ANTONIO MARCOS DE AGUIRRA MASSOLA

AUTOMAÇÃO DE PROJETOS DE SISTEMAS DIGITAIS

SIMULAÇÃO EM NÍVEL DE PORTAS LÓGICAS

Tese de Doutorado apresentada à Escola Politécnica da Universidade de São Paulo, para obtenção do título de Doutor em Engenharia.
Área de Concentração - Engenharia de Eletricidade.

Orientador: Prof. Dr. Antonio Hélio Guerra Vieira

São Paulo
- 1974 -

A minha esposa e a meus filhos
dedico este trabalho.

I N D I C E

S U M Á R I O

Este trabalho apresenta os principais aspectos da automação de projetos de sistemas digitais com a utilização de computadores, detendo-se particularmente no problema da simulação.

Apresenta uma visão do problema geral da automação e suas diferentes alternativas de solução bem como, um relato sobre aspectos da implementação de simuladores.

Introduz um novo algorítmo desenvolvido especificamente para atender às necessidades de simulação em nível de registros e portas. Descreve a implementação de um simulador em nível de portas lógicas, constituido de rotinas escritas em linguagem Assembler e Fortran do sistema HP 2116B do Laboratório de Sistemas Digitais do Departamento de Engenharia de Eletricidade da Escola Politécnica da Universidade de São Paulo.

Finalmente, apresenta as regras de sua utilização, exemplos e as conclusões mais significativas obtidas atráves de seu emprego em vários casos práticos.

A B S T R A C T S

This paper deals with some basic features of computer-aided automation of digital systems design, particularly with those related to the simulation problem.

It presents a general view of this problem and different approuches for solving it and reports also on the main aspects of simulator implementation.

A new algorithm, developed to meet our specific requirements of register-level and gate level simulation, is described, together with the implementation of a gate-level simulator. The latter consists of routines written in the Assembler and Fortran languages of the HP 2116B System and was developed at the Laboratório de Sistemas Digitais do Departamento de Engeharia de Eletricidade da Escola Politécnica da Universidade de São Paulo.

The document also includes rules for the use of the simulator, examples and some relevant conclusions which resulted from the application of the simulator in actual cases.

RÉSUMÉ DE L' AUTEUR

Ce travail présente des aspects principaux de l'automatisation des projets de systèmes digitales, avec l'usage des ordinateurs, particulièrement dans le problème de la simulation.

Il prèsente une vision du problème général de l'automatisation e de ses différentes alternatives de solution, aussi comme un rapport sur des aspects de la réalisation des simulateurs.

Il introduit un nouvel algorithme spécifiquement développé pour pourvoir aux besoins de simulation au niveau de registres et de portes. Il décrit la réalization d'un simulateur au niveau de portes logiques, construit avec des routines écrites dans les languages Assembler et Fortran du système HP 2116B du Laboratório de Sistemas Digitais, Departamento de Engenharia de Eletricidade, Escola politécnica da Universidade de São Paulo.

En concluant, il présente les règles pour son usage, des exemples, et les conclusions plus significatives, obtenues à travers son emploi dans plusieurs cas pratiques.

A G R A D E C I M E N T O S

Ao Professor Doutor Antonio Hélio Guerra Vieira pelo apoio e constante incentivo.

Aos colegas do Laboratório de Sistemas Digitais e em especial ao grupo de "software", pelo apoio e colaboração nas diversas fases do trabalho.

Aos engenheirandos Claudinê Brandão e Haroldo Masakaju Matsumoto, pelo auxílio no software necessário.

Ao engenheirando Paulo Sérgio Moreira, pela preparação dos exemplos de testes.

As secretárias Marilúcia, Celimanna e Augusta pelos serviços de datilografia.

À todos que, direta ou indiretamente, tenham cooperado para a concretização deste trabalho.

1. Introdução

1.1 A importância da simulação em projetos de sistemas digitais.

A cada dia que passa, é maior a utilização de técnicas de simulação em projetos de sistemas digitais.

Êsse fato torna-se cada vez mais evidente bastando-se para isso, analisar as notícias do crescente investimento de fabricantes de sistemas digitais, na área de simulação com a utilização de computadores digitais.

Os primeiros passos de ataque ao problema de simulação em técnicas digitais, reportam-se ao ano de 1956 quando S. R. Cray e R. N. Kish apresentaram o trabalho "A Progress Report on Computer Aplications in Computer Design" o qual encontra-se publicado nos "Proceedings do Western Joint Computer Conference (1956)" páginas de 82 a 85.

É então uma área bastante nova e por isso, os seus principais utilizadores encontram estímulo para aumentar investimentos em pesquisas pois, sendo recentíssima é pouca a documentação disponível sobre o assunto com vistas as suas perspectivas de aplicação, apesar dos problemas encontrados, inúmeros trabalhos já podem ser listados na área, todos traduzindo a vital importância da automação de projetos de sistemas por computadores digitais, nos trabalhos que consomem a maior parte do tempo dos projetistas.

Assim sendo, num projeto de um sistema digital onde são inúmeras as fases, em cada uma delas o uso do computador como ferramenta de trabalho é de primordial importância.

Problemas enfrentados em projeto lógico, partição, rotas de circuito impresso, fiação, testes e documentação, são bem orientados em sua solução quando se

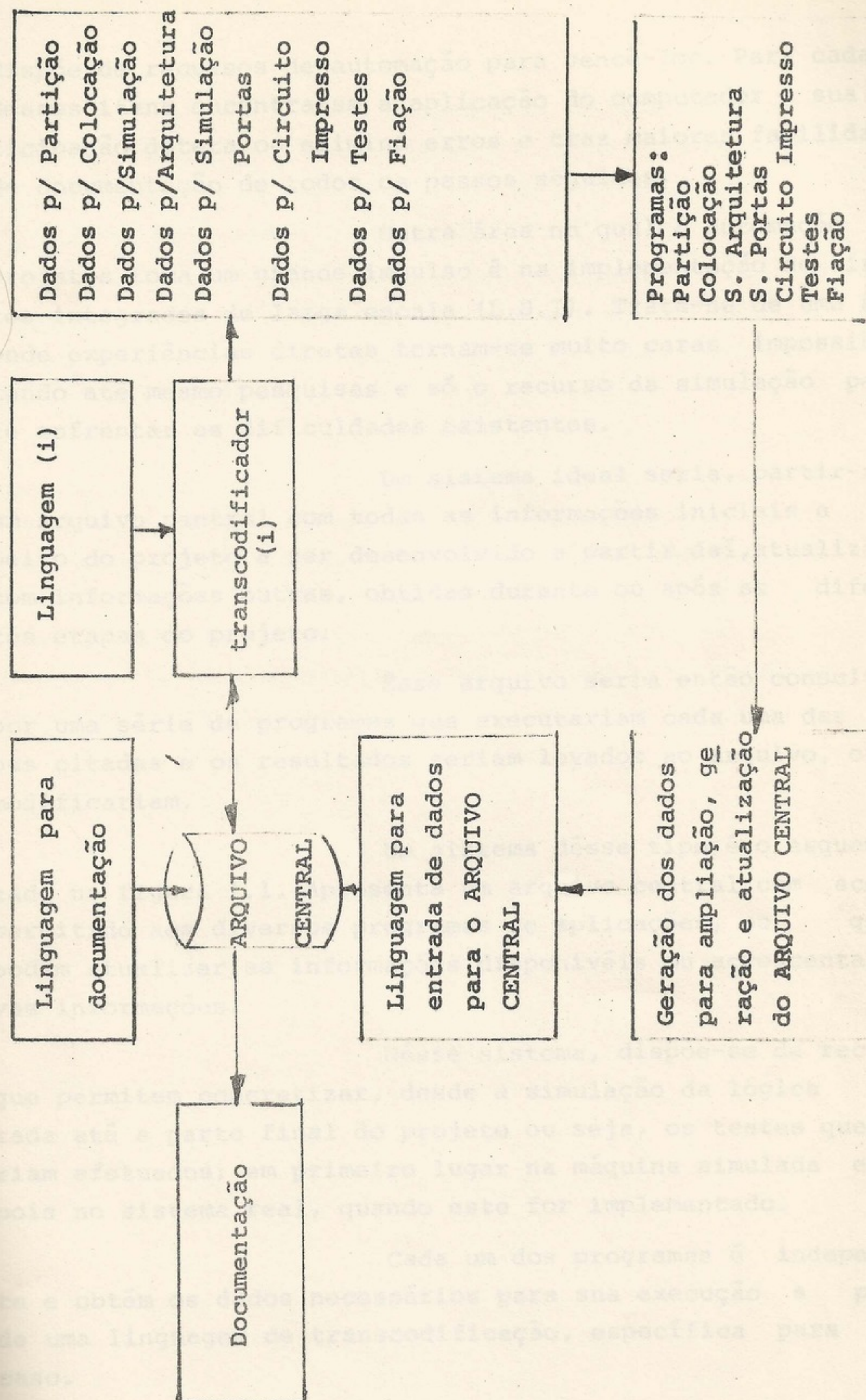



Fig. 1.1 -Estrutura geral de um sistema de automação de projetos

dispõe de recursos de automação para vencê-los. Para cada um desses itens encontra-se a aplicação do computador e sua participação deteta ou elimina erros e traz maiores facilidades de documentação de todos os passos seguidos.

Outra área na qual a automação de projetos toma um grande impulso é na implementação de circuitos integrados de larga escala (L.S.I). Trata-se de uma área onde experiências diretas tornam-se muito caras impossibilitando até mesmo pesquisas e só o recurso da simulação permite enfrentar as dificuldades existentes.

Um sistema ideal seria, partir-se de um arquivo central com todas as informações iniciais a respeito do projeto a ser desenvolvido a partir daí,atualizá-lo com informações outras, obtidas durante ou após as diferentes etapas do projeto.

Êsse arquivo seria então consultado por uma série de programas que executariam cada uma das etapas citadas e os resultados seriam levados ao arquivo, ou o modificariam.

Um sistema dêsse tipo é o esquematizado na figura 1.1. Apresenta um arquivo central com acesso permitido aos diversos programas de aplicações, os quais podem atualizar as informações disponivéis ou acrescentar novas informações.

Nêsse sistema, dispõe-se de recursos que permitem concretizar, desde a simulação da lógica projetada até a parte final do projeto ou seja, os testes que seriam efetuados; em primeiro lugar na máquina simulada e depois no sistema real, quando este for implementado.

Cada um dos programas é independente e obtêm os dados necessários para sua execução a partir de uma linguagem de transcodificação, específica para cada caso.

O programa de documentação encarregar-se-á de estabelecer a atualização de dados disponíveis e produzir listagens dos mesmos. No final do projeto poderá fornecer uma documentação total do sistema desenvolvido.

No presente trabalho foi adotado essa estrutura e implementada uma da suas etapas, ou seja, a simulação lógica em nível de portas, atráves de uma linguagem especialmente desenvolvida.

Embora este trabalho se limite à definição de uma estrutura e à implementação de uma etapa de um todo desejável, outros programas já se encontram em desenvolvimento ou mesmo prontos e farão parte de um sistema único de automação de projetos, que como foi citado, utilizar-se-á de um arquivo de dados único, do qual serão obtidos os dados necessários a execução de cada programa, e que receberá, eventualmente, após a execução, novas informações.

2. Aspectos gerais da simulação lógica e o algorítmo desenvolvido.

2.1 Níveis de simulação lógica

Por simulação lógica entende-se a elaboração de um modêlo do sistema, obedecendo-se a uma determinada estrutura de dados e o seu adequado estímulo a tráves de sinais de testes gerados pelo usuário,obtendo-se assim a resposta do modêlo no decorrer do tempo.

Pode-se atráves de um modêlo do sistema, estudar suas reações e comportamento quando atuando num sistema real, sem necessidade de sua implementação física e consequentes modificações que se mostram necessárias para atingir-se o objetivo desejado.

Os níveis de simulação lógica disponíveis são em número de cinco. A descrição de cada um é apresentado a seguir.

a. Nível de arquitetura ou de sistema

Ao se utilizar êste nível de simulação,tem-se em mente a análise global das propriedades do sistema em estudo. Os elementos que constituem o sistema são na maioria das vezes, dispositivos bastante complexos e devem ser representados por modelos onde os parâmetros principais dizem respeito aos tempos de resposta ou atrasos, capacidade, etc.

Uma lista de dispositivos presentes nêsse nível de simulação é constituida dos seguintes elementos: unidades aritmética e lógica ou simplesmente aritmética, módulos de memória com seus diferentes recursos de implementação: núcleos de ferrite ou semicondutores,portas de seleção, unidades centrais de processamento, decodificador, deslocador, contador, etc.

Cada um desses elementos deve ser modelado convenientemente para permitir sua excitação pelo sistema.

A simulação neste nível é feita considerando-se o "timing" do sistema.

b. Nível de Registros

Para se processar a simulação neste nível, torna-se necessário especificar o fluxo de dados do sistema ao nível de registradores. Neste caso o simulador opera sôbre dados reais e em consequência o sistema pode ser analisado em detalhes e até mesmo programas podem ser executados como se o fossem num sistema real.

Pode-se muitas vezes colocar nêste nível de simulação dispositivos complexos como descrito no ítem a, mas nêste caso devem ter sua descrição e modêlo adequados ao nível de registros.

c. Nível Lógico

Este nível de simulação é indicado quando se deseja efetuar a simulação utilizando-se as equações booleanas que descrevem o sistema. Para isso utiliza-se apenas os dois níveis lógicos "0" e "1" e o tempo unitário de simulação corresponde ao sinal do relógio central do sistema. Esta simulação é utilizada exclusivamente quando se deseja verificar a lógica do sistema.

d. Nível de Portas

Quando se dispõe da descrição do sistema atráves de portas lógicas simples ("gates") interconectadas, utiliza-se a simulação em nível de portas. Neste caso cada porta deverá ter um modelo bastante real, com parametros indicativos dos atrasos envolvidos, "fan-in" e "fan-out", etc.

e. Nível de Circuitos

As portas descritas para o simulador em nível de portas podem ser constituidas de interconecções de resistências, transistores, diodos e condensadores. A sua simulação se desejada nêste nível, será considerada como a nível de circuitos e neste caso os dados não se limitarão aos valores lógicos "0" ou "1" mas sim a valores quantitativos que correspondem à indicações de tensão ou corrente.

O modêlo para este nível de simulação consiste dos circuitos equivalentes dos dispositivos descritos acima.

2.2 - Características principais associadas a um sistema de simulação.

A escolha de uma eficiente estrutura de dados é de grande significado para o sistema que se deseja implementar.

Para se construir a estrutura sobre a qual o simulador operará, duas alternativas são apresentadas ao projetista do programa. A primeira será construí-la a partir de uma linguagem de entrada do tipo compilador, onde, a medida em que se recebe as informações estas serão convertidas em chamadas de sub-programas especifícos que executam as funções representadas por um bloco lógico. As chamadas de sub-programas gerados deverão ser interligadas a fim de permitir a descrição lógica da rêde. Após essa montagem, o sistema irá proceder a transformação dessas chamadas em código objeto de máquina para posterior execução.

Nêste caso, o sistema mostrar-se-á pouco eficiente pois se houver atualizações frequentes no projeto estas implicam em atualizar a estrutura atráves de uma nova passagem pela linguagem de entrada e de toda a descrição do sistema, pois um novo código objeto deverá ser obtido.

Uma segunda alternativa será partir para a obtenção de uma estrutura numérica independente de código objeto de máquina de modo a permitir que, havendo mudanças de projeto estas possam ser fàcilmente absorvidas pelo sistema, sem necessidade de toda a rede sofrer uma nova passagem pela linguagem de entrada mas sim, apenas as modificações.

Neste caso a linguagem de entrada obtém as informações para o sistema e as coloca na estrutura como se fôsse preencher uma tabela. O simulador ao ser executado, irá percorrer essa estrutura e a medida que for necessitando dos dados os interpretará para a realização da simulação. Esse fato permite considerar o simulador como interpretador, pois é a interpretação dos dados, sua principal característica.

A estrutura de dados gerada dessa maneira permite ainda a consideração de outros detalhes (como por exemplo de que pino procede um sinal)bastando-se para isso apenas sua configuração na estrutura. O sistema ainda é acessível para testes de comportamento, durante a fase de desenvolvimento pois, uma tabela é mais fácil de ser interpretada que um código objeto.

A segunda alternativa foi a adotada.

Outro ponto a discutir é a escolha do tipo de simulador a ser desenvolvido.

Um simulador, será denominado sincrono quando todos os elementos que constituem sua estrutura podem ter seus valores utilizados, apenas a intervalos regulares de tempo, caracterizados pela presença de um sinal de relógio. Assim sendo, os valores assumidos pelo elemento fora desses intervalos padrão, não terão significados para o usuário.

Se entretanto, os valores dos sinais que constituem a estrutura puderem ser utilizados tão logo ocorra uma mudança de estado, independendo de intervalos regulares de tempo controlado pelo simulador, ele será denominado assincrono.

Isso entretanto não implica que certos simuladores, ditos sincronos só possam ser utilizados por sistemas sincronos e os assincronos por sistemas assincronos.

Dentre as características associadas a um bloco, uma das principais a ser considerada será o atraso a ele associado. Muitas vezes, como é comum ocorrer, um único atraso não é característica principal de um bloco, pois dependendo do projeto lógico do qual o bloco participa, torna-se necessário levar outros atrasos em consideração.

De uma maneira geral o que se considera, é o atraso de propagação que caracteriza o bloco ao variar sua saída do valor lógico "0" para "1", que será denominado TD01 e o atraso correspondente ao bloco mudar do nível lógico "1" para "0", que será denominado TD10.

Esses atrasos são sempre valores constantes e normalmente o que se utiliza, são valôres nominais dos atrasos fornecidos pelo próprio fabricante.

As demais características dos blocos lógicos e que são utilizadas no programa simulador, dizem respeito aos problemas de "fan-in" e "fan-out", bem como, as condições de testes de sinais de dados, para determinados blocos lógicos, representados pelos tempos de "hold" e de "set-up".

Para se definir o que é "fan-in" ou "fan-out" como utilizado no programa, deve-se antes de mais nada definir-se o que é unidade de carga.

Unidade de Carga é considerada como uma entrada simples de uma porta lógica de uma família qualquer. Isso significa para o caso da família TTL - 7400, uma carga representada pela corrente de menos 1.6 m A quando no estado lógico "0" e de 40 μA quando no estado lógico "1".

"Fan-in" de um bloco lógico é o número obtido pela soma das unidades de carga de todas as entradas do bloco em questão.

"Fan-out" de uma saída, é o número máximo de unidades de carga que podem ser ligadas a essa saída, assegurando os níveis "0" e "1".

O tempo de "hold" é definido como sendo, o intervalo de tempo em que as entradas correspondentes a dados, devem permanecer estáveis, após o pulso de "clock" ter caído abaixo de um determinado valor (ordem 50% ou então 1.5V no caso da família TTL) para assegurar operação correta do "flip-flop".

O tempo de "set-up" é definido como sendo, o intervalo de tempo em que as entradas correspondentes a dados, devem permanecer estáveis, antes da ação do pulso de "clock" no "flip-flop".

A figura ilustra, com a forma de onda, os tempos de "hold" e "set-up".

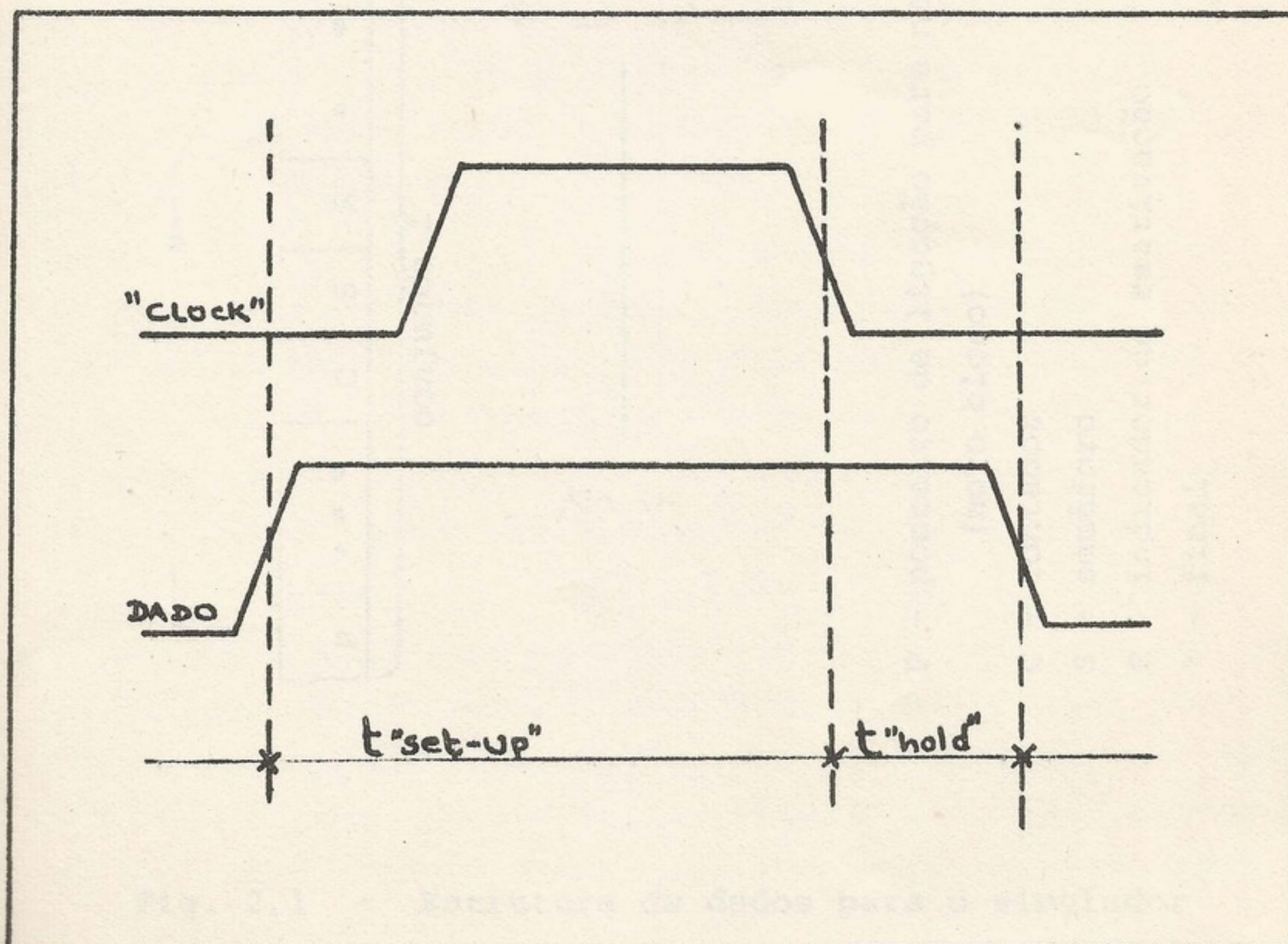
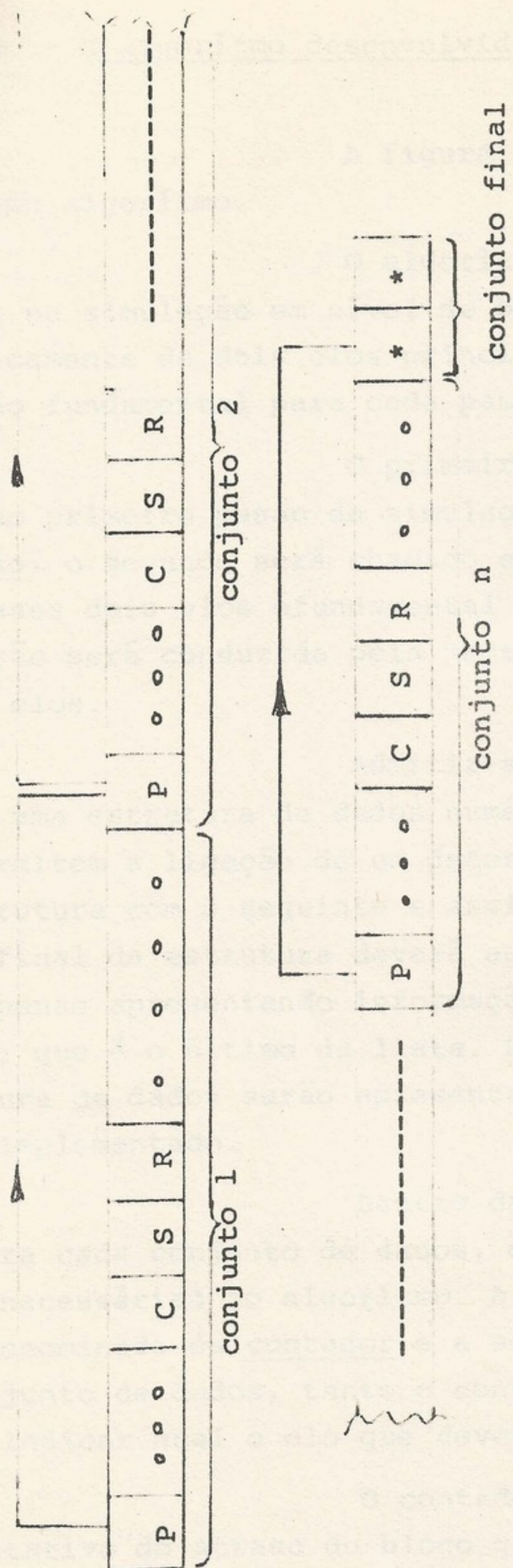



Fig. 2.1 - Estrutura de dados para o simulador

2.3 - O algorítmo desenvolvido para o sistema

A figura 2.2 ilustra o diagrama em blocos do algorítmo.

O algorítmo desenvolvido para utilização na simulação em nível de portas e de registros,consta basicamente de dois elos principais. Cada elo representa uma ação fundamental para cada passo da simulação.

O primeiro elo, que também corresponde ao primeiro passo da simulação, é chamado de elo da execução; o segundo será chamado elo da ativação. A existência desses dois elos éfundamental para o algorítmo e toda simulação será conduzida pela passagem obrigatória atráves desses elos.

Admitir-se-á por ora, que se dispõe de uma estrutura de dados numérica, atráves de listas que permitem a ligação de um determinado conjunto de dados da estrutura com o seguinte e assim sucessivamente. O conjunto final da estrutura deverá apresentar mesma organização, apenas apresentando informação adicional que permita indicar que é o último da lista. Detalhes a respeito dessa estrutura de dados serão apresentados na descrição do simulador implementado.

Dentro da estrutura em questão,existem para cada conjunto de dados, duas posições bem definidas e necessárias ao algorítmo. A primeira dessas posições será denominada de contador e a segunda de semáforo. Em cada conjunto de dados, tanto o contador como o semáforopermitarão indicar qual o elo que deverá ser processado.

O contador deverá ter conteúdo representativo do atraso do bloco que representa na estrutura, e o semáforo será apenas o indicador de atividade para o bloco representado na estrutura. Pode-se verificar a disposição dos elementos citados, na figura 2.1. Apresenta-se ainda, uma outra posição denominada indicador de reativação,cuja finalidade será descrita quando de sua utilização na descrição do algorítmo.

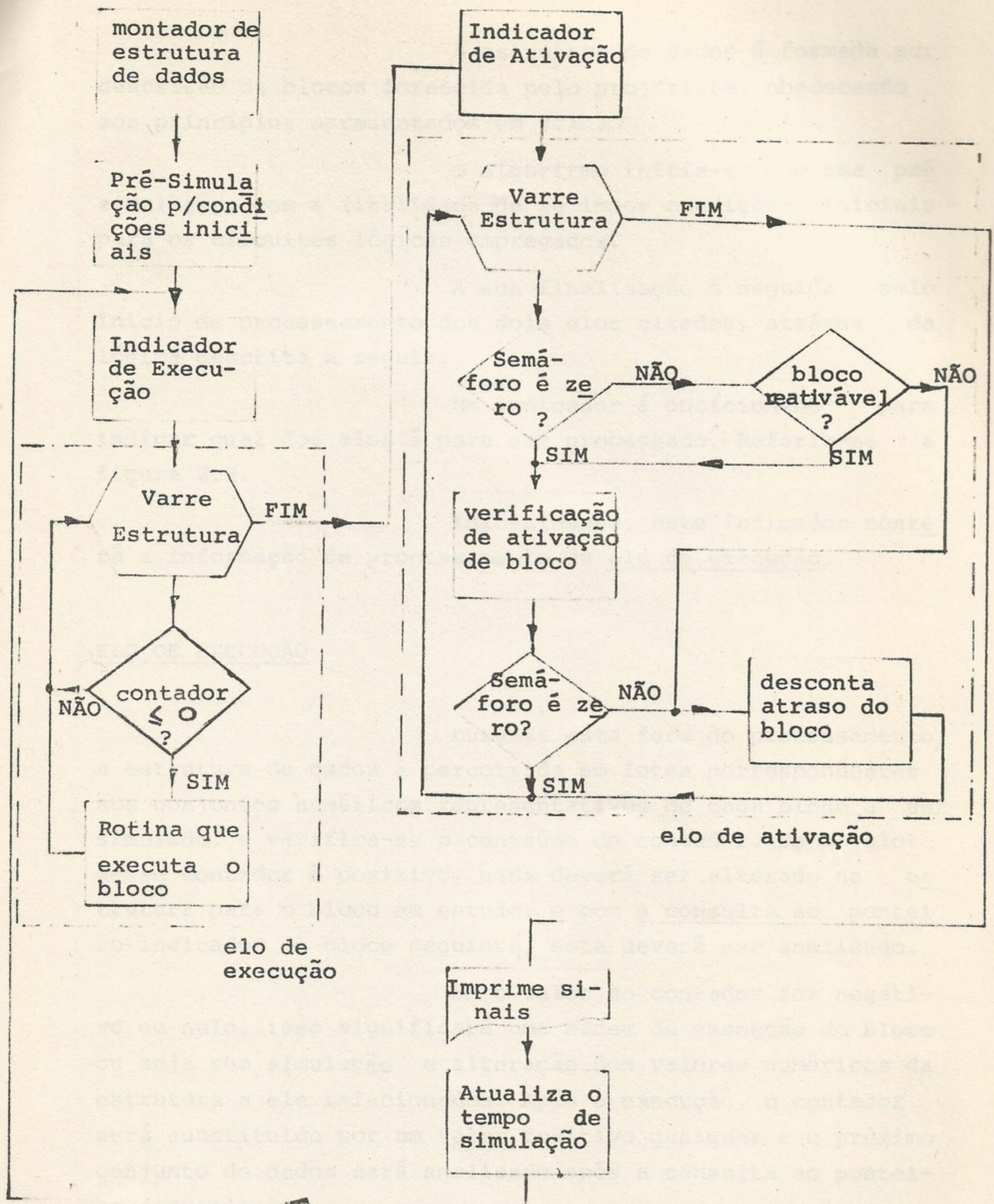



Fig. 2.2 - Diagrama em blocos do algorítmo desenvolvido

A estrutura de dados é formada por descrição de blocos fornecida pelo projetista, obedecendo aos principios apresentados em 2.1 .

O algoritmo inicia-se com uma pré simulação, com a finalidade de se impor condições iniciais para os circuitos lógicos empregados.

A sua finalização é seguida pelo ínicio de processamento dos dois elos citados, atráves da lógica descrita a seguir.

Um indicador é posicionado para indicar qual dos elos é para ser processado. Referir-se a figura 2.2.

Inicialmente, este indicador conterá a informação de processamento do elo de execução.

ELO DE EXECUÇÃO

Durante esta fase do processamento, a estrutura de dados é percorrida em lotes correspondentes aos conjuntos numéricos representativos de cada bloco a ser simulado, e verifica-se o conteúdo do contador. Se o valor desse contador é positivo, nada deverá ser alterado na estrutura para o bloco em estudo, e com a consulta ao ponteiro indicador do bloco seguinte, este deverá ser analisado.

Se o valor do contador for negativo ou nulo, isso significará uma ordem de execução do bloco ou seja sua simulação e alteração dos valores numéricos da estrutura a ele relacionados. Após a execução, o contador será substituido por um valor positivo qualquer e o próximo conjunto de dados será analisado após a consulta ao ponteiro conveniente.

Chegando-se ao fim da estrutura de dados, o indicador que estava posicionado para comandar o elo de execução terá seu conteúdo indicando agora o elo de ativação e esse será agora processado.

ELO DE ATIVAÇÃO

Durante esta fase, novamente a estrutura será percorrida pelo simulador. Para cada conjunto numérico é agora efetuada a análise do semáforo.

Dessa análise, resultam duas alternativas possíveis:

o conteúdo do semáforo é zero

Neste caso o algorítmo deve verificar se o bloco em estudo deverá ser ativado. Esta verificação é efetuada atráves do estudo dos estados dos sinais de entrada do bloco e do seu valor atual de saída. Se houver em decorrência, uma necessidade de mudança na saída do bloco, o mesmo deve ser ativado. Isso significa colocar-se no contador o valor do atraso a ser descontado e no semáforo o valor igual a 1 (ativado). Não havendo causa que justifique a ativação,do semáforo, este permanecerá com seu valor igual a 0.

o conteúdo do semáforo não é zero

Neste caso, apesar do bloco estar ativado por alguma causa anterior (por exemplo, já descontando o contador) deve-se verificar se o mesmo é reativável, o que é possível atráves da consulta à posição R correspondente à informação de reativação. A figura 2.1 indica onde está esta posição.Se for reativável, deve-se verificar se o estado atual das entradas provoca a variação no valor da saída do bloco sendo simulado. Se isso realmente acontece, o bloco deve ter seu contador novamente reposicionado com o valor do atraso correspondente ao caso em questão. Caso contrário, o processamento deve prosseguir sem alteração no conteúdo do contador.Se o bloco não é reativável, deve-se prosseguir o processamento, sem alteração no conteúdo do contador permanecendo este com o valor e que deverá ser descontado no momento oportuno.

Em ambos os casos, quer o semáforo contenha ou não zero, o processamento deve prosseguir com a análise do próximo bloco da estrutura.

Durante a fase de teste do semáforo, deve-se descontar uma unidade de atraso do valor existente no contador, para os blocos que estiverem ativados.

O elo de ativação termina quando a consulta à estrutura indicar seu final; neste caso, o algorítmo deve fornecer as informações correspondentes à situação dos blocos simulados e atualizar o tempo de simulação.

A simulação, se necessário prosseguir, continuará com o elo de execução, pois o indicador correspondente a esta atividade foi posicionado após o término do elo de ativação.

3. O programa para simulação em nível de portas.

3.1 -Descrição Geral

O sistema implementado consta basicamente de duas partes principais, com funções bem distintas e que quando executadas, fornecem informações básicas para o sistema ou a própria simulação do mesmo.

A figura 3.1 ilustra o esquema geral do programa de simulação em nível de portas lógicas.

As duas partes citadas são:

a) linguagem de entrada,para leitura de dados e montagem da estrutura;

b) simulador propriamente dito.

A linguagem de entrada permite que dados, a serem utilizados para a montagem da estrutura de dados do simulador, sejam recebidos e armazenados na estrutura de uma maneira conveniente.

Entenda-se como linguagem de entrada, não uma linguagem formal estruturada, mas sim, um programa preparador de dados para a estrutura a ser montada.

Sua função básica é, então, a leitura de dados, escolha de controle a executar, montagem da estrutura de dados e indicar se existem ou não erros nas informações fornecidas, permitindo, com a impressão do numéro do erro cometido, a ação do operador para sua orientação e correção.

O simulador propriamente dito é constituído de várias rotinas que deverão correr toda estrutura montada, modificando-a convenientemente e de acôrdo com os passos seguidos na simulação.

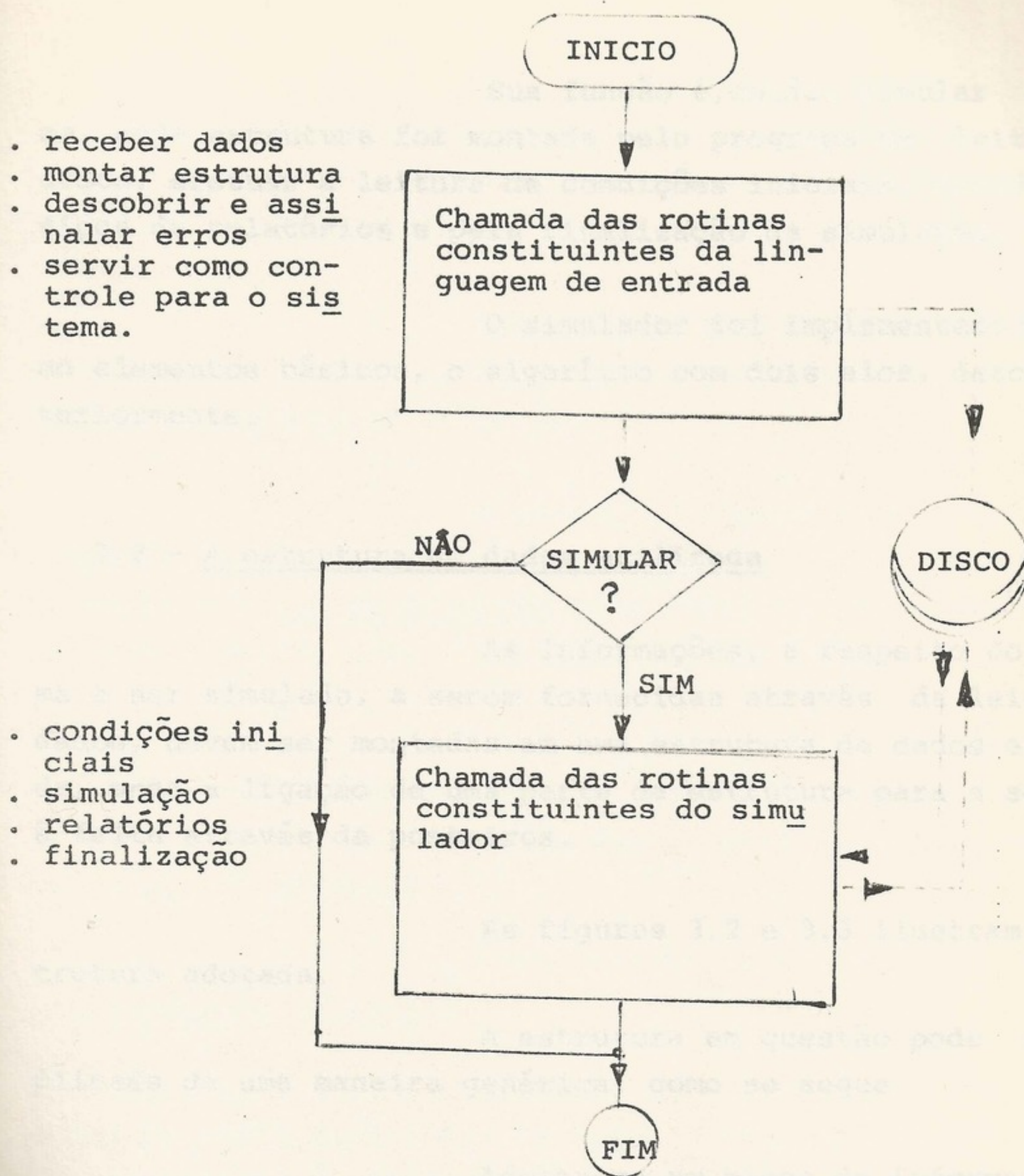



Fig. 3.1 - Esquema Geral do programa de simulação à nível de portas lógicas

Sua função é,então, simular o sistema, cuja estrutura foi montada pelo programa de leitura de dados, efetuar a leitura de condições iniciais, decidir por tipos de relatórios e pela finalização da simulação.

O simulador foi implementado tendo como elementos básicos, o algorítmo com dois elos, descrito anteriormente.

3.2 - A estrutura de dados utilizada

As informações, a respeito do sistema a ser simulado, a serem fornecidas através da leitura de dados, devem ser montadas em uma estrutura de dados encadeada, onde a ligação de uma parte da estrutura para a seguinte é feita através de ponteiros.

As figuras 3.2 e 3.3 ilustram a estrutura adotada.

A estrutura em questão pode ser explicada de uma maneira genérica, como se segue.

Admita-se um bloco de informação. A primeira palavra deste bloco será destinada ao ponteiro indicador do próximo bloco; no caso em estudo, será a primeira posição livre da estrutura de dados, localizada após a completa descrição do bloco presente. Isso mostra que essa primeira posição só será preenchida após o processamento de todas as informações do bloco em questão.

A segunda palavra é reservada ao número característico da rotina relacionada com a descrição do bloco.

Cada rotina terá um único número, o qual permitirá sua chamada, quer no elo de execução quer no elo de ativação do algorítmo utilizado, quando da exploração da estrutura pelo simulador.

Esse número é utilizado, também,para caracterizar o bloco como sendo uma porta lógica,um bloco lógico, ou um bloco de controle.

A terceira palavra é reservada ao contador de atrasos, ou seja, a posição que deverá ser consultada durante o elo de execução para verificar se o bloco deve ser executado ou não.

A quarta palavra é destinada ao semáforo. É esta posição que deverá indicar se o bloco foi ativado ou não e, de sua consulta, dependerá toda lógica existente no elo de ativação.

Esta posição será também chamada de indicadora de ativação.

A quinta palavra é destinada ao indicador de reativação. Esta informação permitirá a verificação da necessidade de se recomeçar a descontar o atraso, quando a variação de sinais de entrada do bloco provoca um sinal de saída diferente daquele que está sendo simulado. Este indicador só é testado quando o bloco já estiver ativado. As palavras sucessivas da estrutura serão utilizadas para guardar os atrasos típicos do bloco, número e respectivos sinais,função do bloco, áreas de rascunhos, nome do sinal de saída do bloco, etc. A figura 3.2 mostra o significado de cada palavra de um bloco da estrutura, enquanto a figura 3.3 ilustra o aspecto da estrutura preenchida pelos dados dos dois blocos lógicos, apresentados na própria figura.

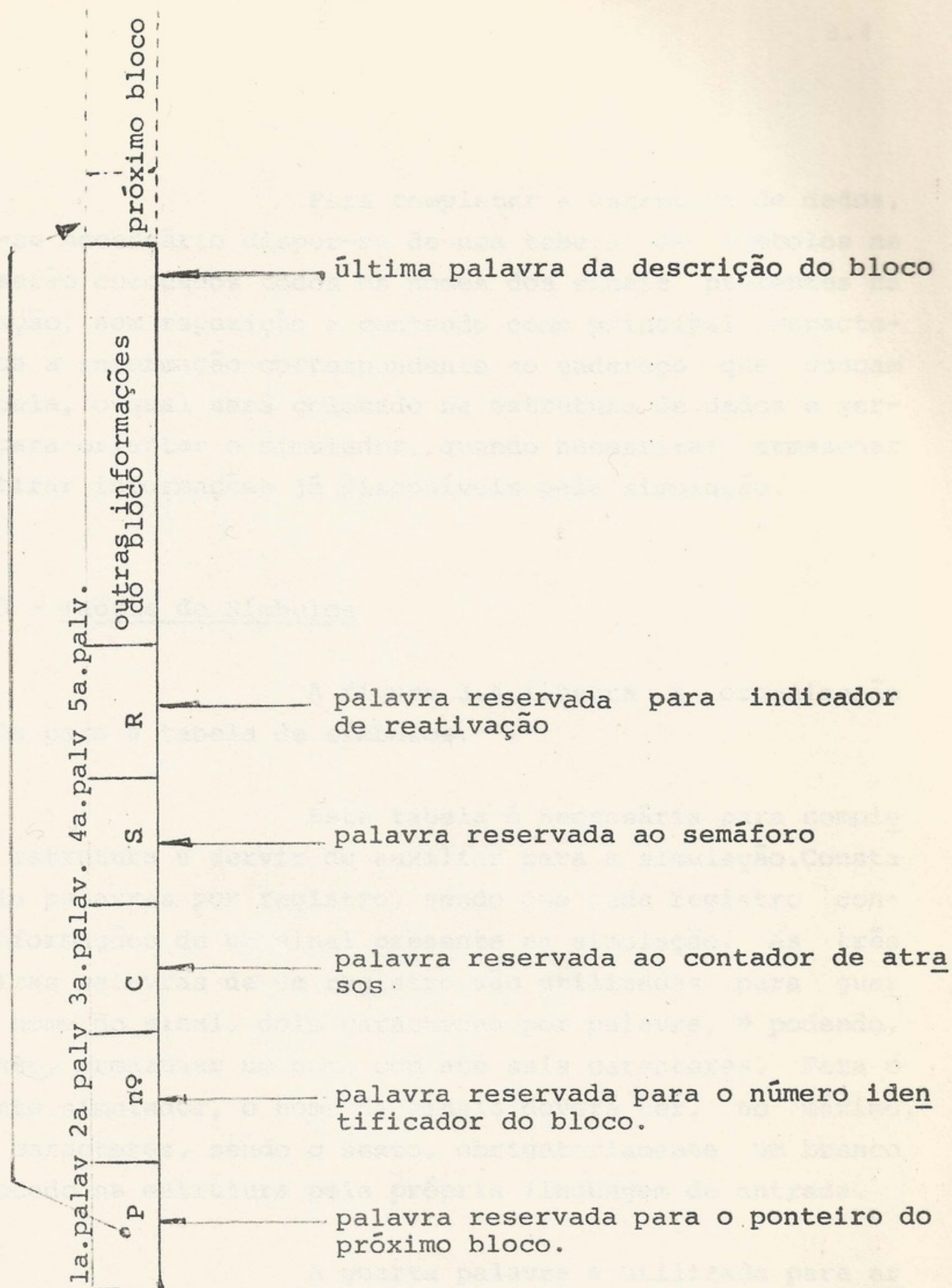



Fig. 3.2 - Composição de um bloco típico

Para completar a estrutura de dados, torna-se necessário dispor-se de uma tabela de símbolos na qual serão colocados todos os nomes dos sinais presentes na simulação, sem repetição e contendo como principal característica a informação correspondente ao endereço que ocupam na tabela, o qual será colocado na estrutura de dados e servirá para orientar o simulador, quando necessitar armazenar ou retirar informações já disponíveis pela simulação.

3.3 - Tabela de Símbolos

A figura 3.4 ilustra a organização adotada para a tabela de símbolos.

Esta tabela é necessária para completar a estrutura e servir de auxiliar para a simulação.Consta de seis palavras por registro, sendo que cada registro contém informações de um sinal presente na simulação. As três primeiras palavras de um registro são utilizadas para guardar o nome do sinal, dois caracteres por palavra, e podendo, portanto, armazenar um nome com até seis caracteres. Para o presente simulador, o nome de sinais deverá ter, no máximo, cinco caracteres, sendo o sexto, obrigatoriamente um branco e colocado na estrutura pela própria linguagem de entrada.

A quarta palavra é utilizada para armazenar o endereço do sinal e que é utilizado para ser colocado na estrutura de dados quando a referência ao mesmo fôr efetuada. A figura 3.3 ilustra o fato.

A quinta palavra serve para armazenar o número que caracteriza o tipo do bloco lógico, ao qual o sinal está ligado.

blocos lógicos identificados por:

número 1
não reativável
função W
atrasos TD01
TD10

número 2
reativável
função L
atrasos TD01
TD10

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	18	18	19	20	21	22	23	24
12	1	C	S	O	•	TD01	TD10	3	•	•	•	23	2	C	S	1	•	TD01	TD10	2	•	•	..	..

Tabela de Símbolos

nome do sinal	endereço
T	1
X	2
Y	3
Z	4
A	5
B	6

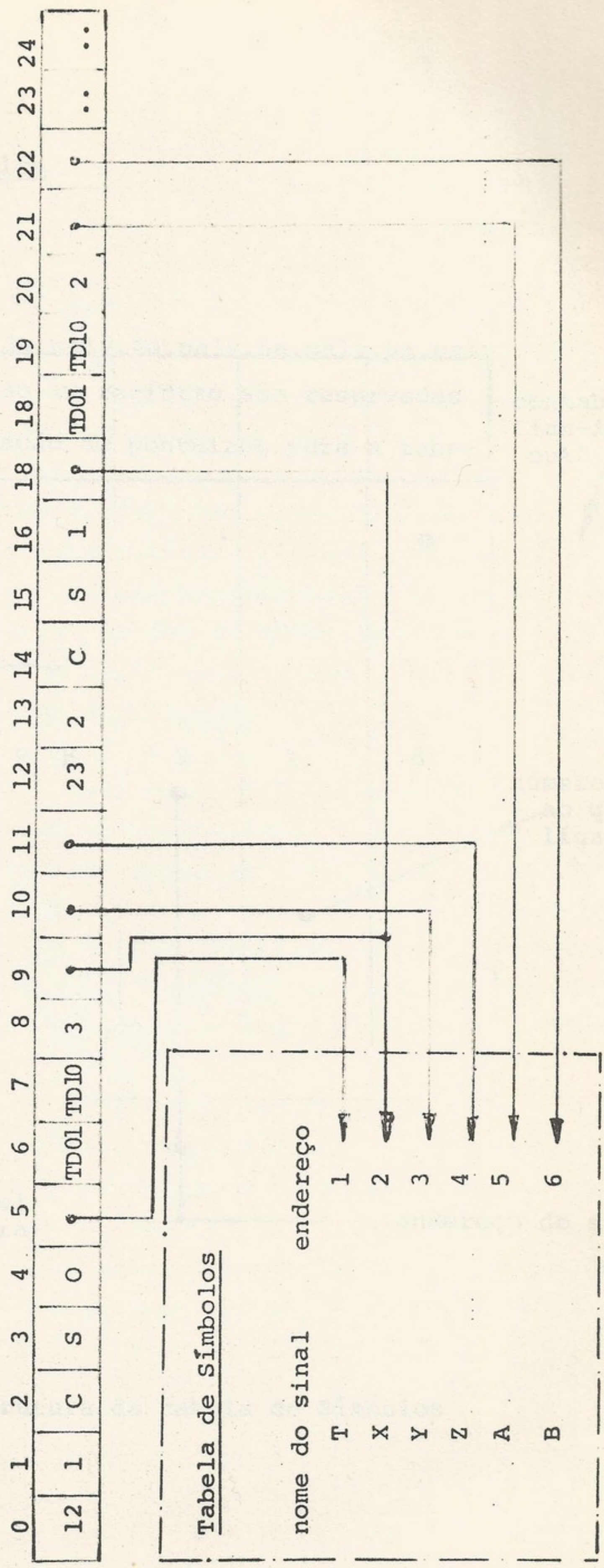



Fig. 3.3 - Estrutura de dados

Tabela de Símbolos

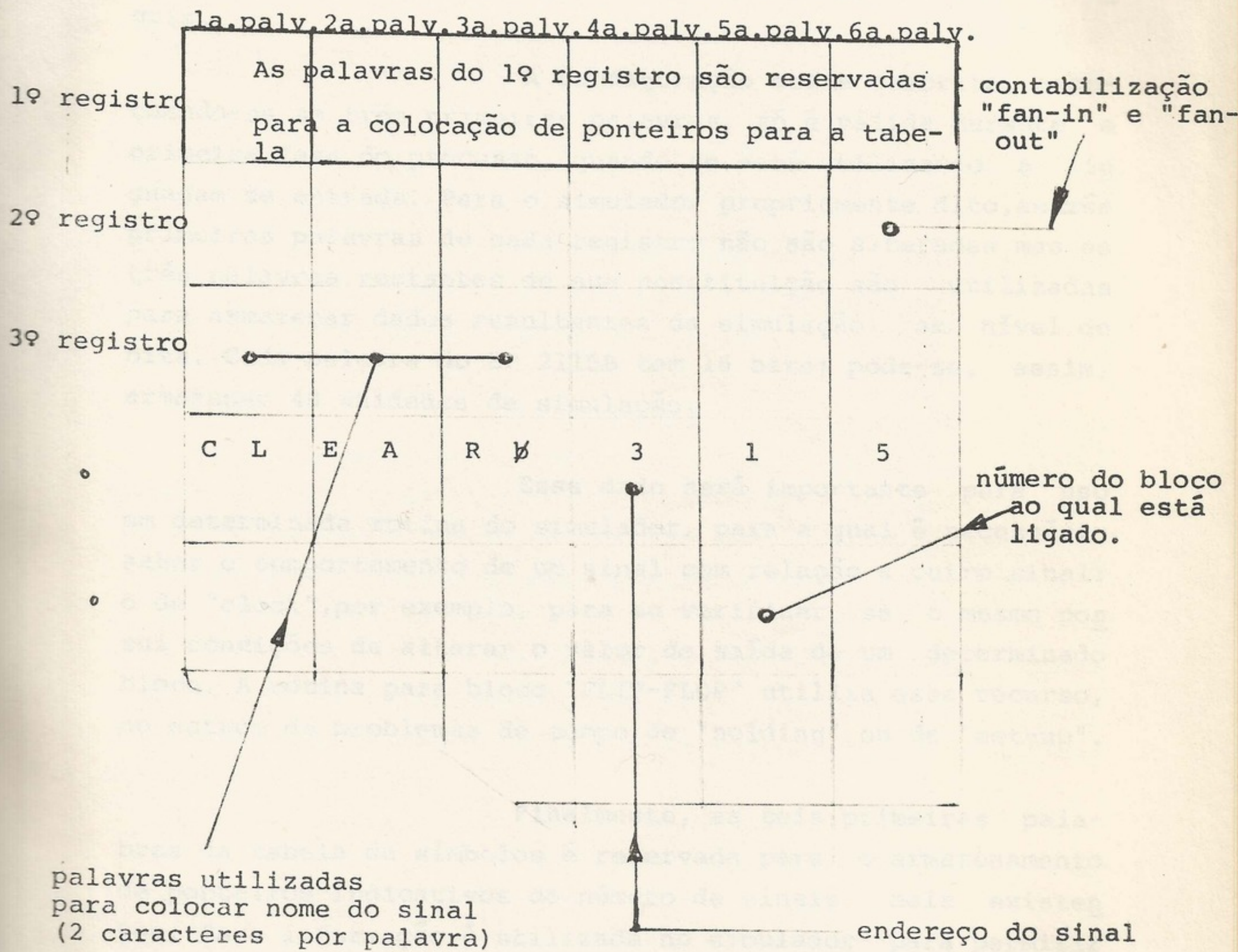



Fig. 3.4 - Estrutura da tabela de Símbolos

A sexta palavra é utilizada para registros de variações de "fan-in" ou "fan-out" do bloco, do qual o sinal procede ou está se dirigindo. O comportamento do circuito quanto ao "fan-in" ou "fan-out" de cada bloco é motivo de um algorítmo específico e que se apresenta a seguir.

A configuração acima descrita, excetuando-se as três primeiras palavras, só é válida durante a primeira fase do processo, quando se está utilizando a linguagem de entrada. Para o simulador propriamente dito, as três primeiras palavras de cada registro não são alteradas mas as três palavras restantes de sua constituição são utilizadas para armazenar dados resultantes da simulação, em nível de bits. Cada palavra do HP 2116B tem 16 bits; pode-se, assim, armazenar 48 unidades de simulação.

Esse dado será importante para uso em determinada rotina do simulador, para a qual é necessário saber o comportamento de um sinal com relação a outro sinal: o de "clock", por exemplo, para se verificar se o mesmo possui condições de alterar o valor de saída de um determinado bloco. A rotina para bloco "FLIP-FLOP" utiliza esse recurso, no estudo de problemas de tempo de "holding" ou de "set-up".

Finalmente, as seis primeiras palabras da tabela de símbolos é reservada para o armazenamento de ponteiros indicativos do número de sinais nela existentes. Essa informação é utilizada no simulador para permitir quer o estabelecimento de condições iniciais para sinais, como também para se saber até onde efetuar a pesquisa na tabela para reconhecimento de sinais.

3.4 - Algorítmo para verificação de "fan in" e"fan-out"

A necessidade representada pela verificação das condições de "fan-in" e "fan-out" de cada bloco,

levou a elaboração de um algorítmo que efetua, automaticamente, a contabilização dos requisitos de correntes de cada sinal que se liga a um dado bloco, bem como para os sinais que, partindo de um dado bloco, se dirige para outro.

O algorítmo elaborado é bem simples e mostrou-se eficiente para solucionar o problema em questão.

A figura 3.5 é o diagrama em bloco do algorítmo.

Consta de três partes bem distintas e devem estar presentes no programa que monta a estrutura de dados, ou seja, na linguagem de entrada.

Na primeira parte do algorítmo, efetua-se a colocação de zeros em todas as posições que correspondam à sexta palavra dos registros. Assim, toda a tabela de símbolos tem a sexta palavra com zeros, independentemente do número de sinais que comparecerão na simulação.

A segunda parte do algorítmo é utilizada durante o processamento dos dados que constituem informações para a montagem da estrutura principal, que será utilizada no simulador.

Nesse ponto, o algorítmo deverá decidir para cada sinal presente no bloco se é uma entrada para o mesmo ou saída,e atualizar a posição seis do registro correspondente ao sinal em questão, somando-se o valor de "fan-out" se o sinal representar uma saída, ou subtraindo-se o valor de "fan-in" se o sinal fôr uma entrada.

Existem, entretanto, sinais que procedem de blocos apenas auxiliares na simulação para os quais não há sentido em falar de problemas relacionados com "fan-in"

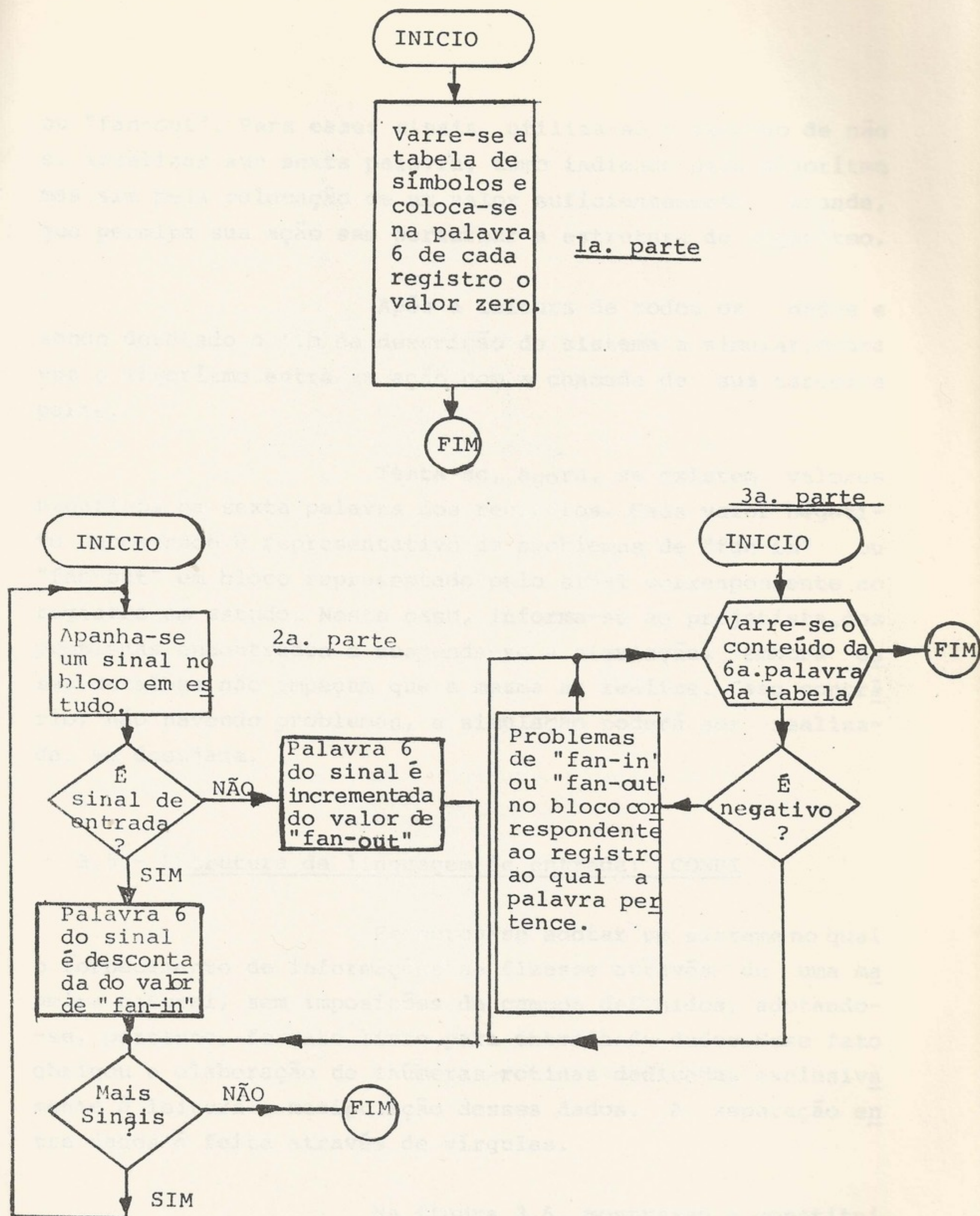



Fig. 3.5 - Algorítmo de "fan-in" e "fan-out"

ou "fan-out". Para esses sinais, utiliza-se o recurso de não se atualizar sua sexta palavra, como indicado pelo algorítmo, mas sim pela colocação de um valor suficientemente grande, que permita sua ação sem perturbar a estrutura do algorítmo.

Após a leitura de todos os dados e sendo detetado o fim da descrição do sistema a simular, outra vez o algorítmo entra em ação com a chamada de sua terceira parte.

Testa-se, agora, se existem valores negativos na sexta palavra dos registros. Cada valor negativo encontrado é representativo de problemas de "fan-in" ou "fan-out" em bloco representado pelo sinal correspondente ao registro em estudo. Neste caso, informa-se ao projetista, dos problemas encontrados e suspende-se a simulação, embora esses detalhes não impeçam que a mesma se realize. Caso contrário, não havendo problemas, a simulação poderá ser realizada, se desejada.

3.5 - Estrutura da linguagem de entrada: COMPI

Procurou-se adotar um sistema no qual o fornecimento de informações se fizesse através de uma maneira natural, sem imposições de campos definidos, adotando-se, portanto, formato livre para entrada de dados. Esse fato obrigou a elaboração de inúmeras rotinas dedicadas exclusivamente à leitura e manipulação desses dados. A separação entre dados é feita através de vírgulas.

Na figura 3.6, mostra-se a constituição genérica de um cartão de dados, bem como os códigos identificadores de blocos lógicos ou de controles, e na figura 3.7, apresenta-se o diagrama em blocos do programa que constitui a linguagem de entrada.

formato gênerico

CB, nome do bloco, { outras informações a respeito do bloco },,

onde CB é o código identificador do bloco ou de controle.

códigos de bloco: A serem colocados como os dois primeiros caracteres de um cartão.

CL - "clock" normal
CR - "clock" reverso
GP - gerador de sinais
GT - bloco porta lógica ("Gate")
FF - bloco "Flip-Flop"
GB - guardar bloco
BL - montar bloco
DB - apagar bloco

códigos de controle: A serem colocados como os dois primeiros caracteres de um cartão.

** - comentário
DI - guardar no disco
LI - Listar a estrutura montada
LB - Listar blocos gravados
FI - final de estrutura
TE - término de uso da linguagem de entrada
NB - novo bloco

Fig. 3.6 - Cartão genérico e códigos

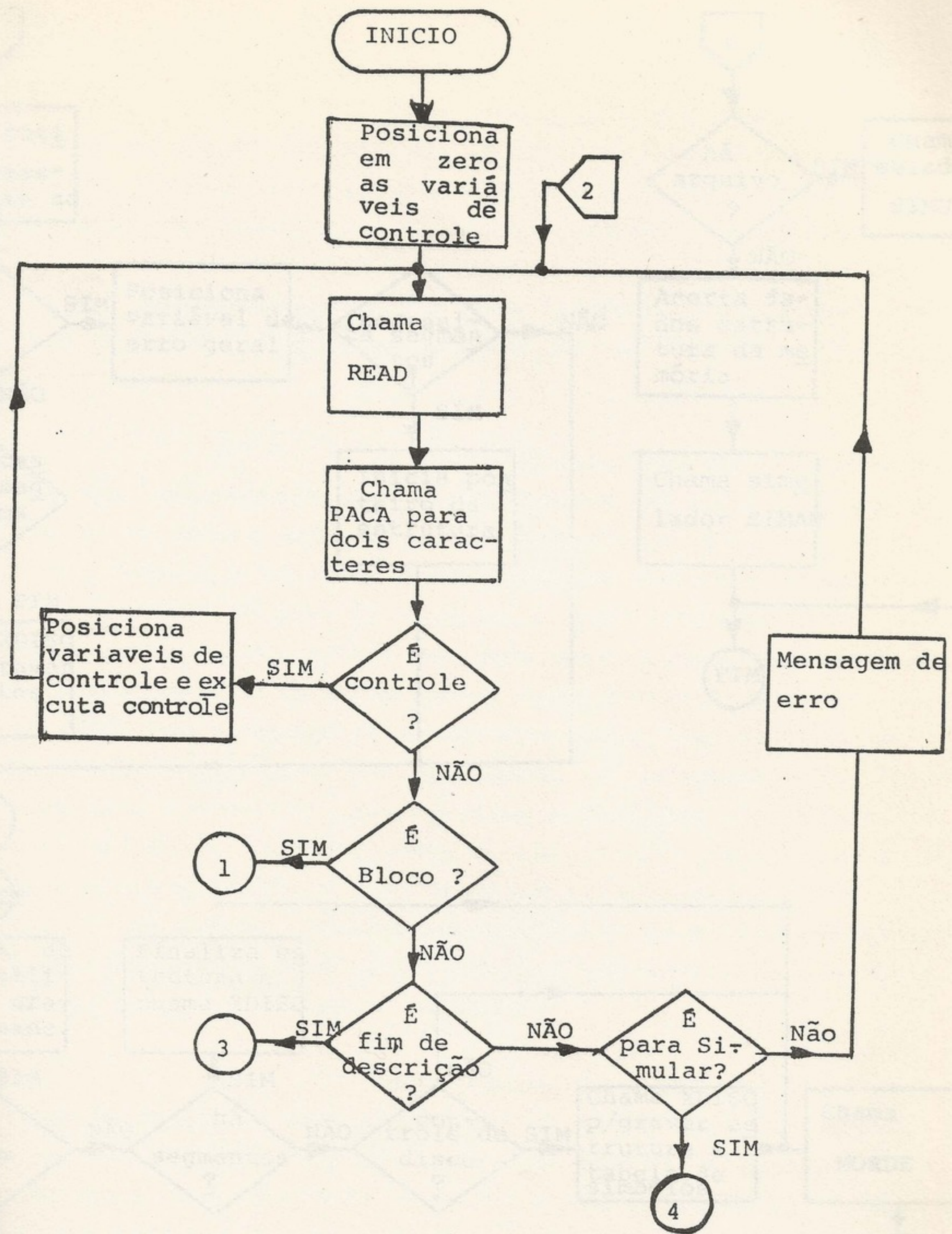



Fig. 3.7 - Linguagem de entrada - Esquema Geral

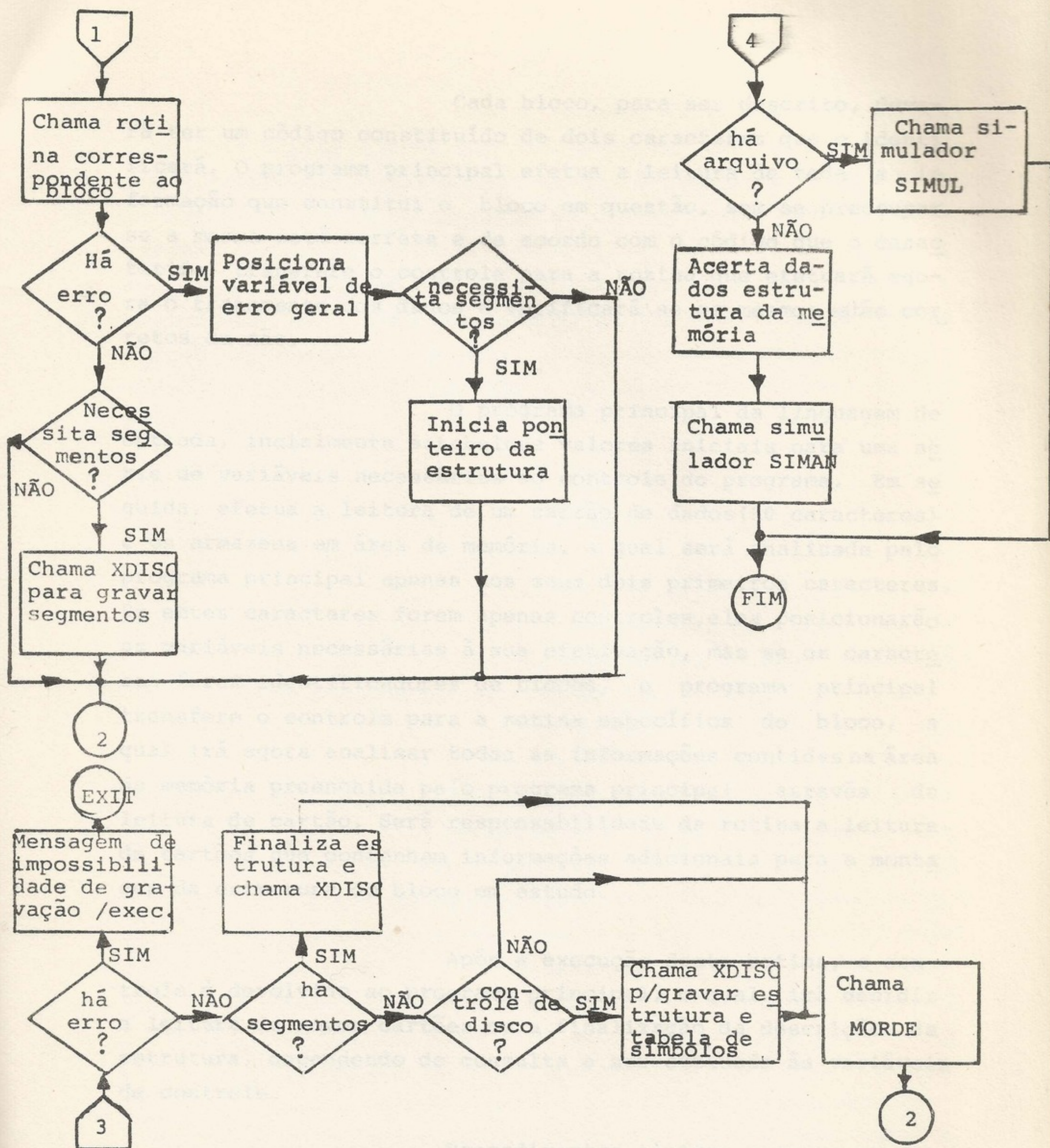



Fig. 3.7 - continuação

Cada bloco, para ser descrito, deverá ter um código constituído de dois caracteres que o identificará. O programa principal efetua a leitura de toda a informação que constitui o bloco em questão, sem se preocupar se a mesma está correta e,de acordo com o código que o caracteriza, transfere o controle para a rotina que efetuará agora o tratamento dos dados e verificará se os mesmos estão corretos ou não.

O programa principal da linguagem de entrada, incialmente estabelece valores iniciais para uma série de variáveis necessárias ao controle do programa. Em seguida, efetua a leitura de um cartão de dados(80 caracteres) e os armazena em área de memória, a qual será analisada pelo programa principal apenas nos seus dois primeiros caracteres. Se estes caracteres forem apenas controles,eles posicionarão as variáveis necessárias à sua efetivação, mas se os caracteres forem identificadores de blocos, o programa principal transfere o controle para a rotina específica do bloco, a qual irá agora analisar todas as informações contidas na área de memória preenchida pelo programa principal através da leitura de cartão. Será responsabilidade da rotina a leitura de cartões que contenham informações adicionais para a montagem da estrutura do bloco em estudo.

Após a execução desta rotina, o controle é devolvido ao programa principal, o qual irá decidir a leitura de novos cartões ou a finalização da descrição da estrutura, dependendo de consulta a ser efetuada às variáveis de controle.

Procedimentos típicos a serem executados pelo programa principal, quando da análise das variáveis de controle serão, por exemplo: a necessidade de armazenar no disco a estrutura existente na memória, segmentar a estrutura com armazenamento em disco quando houver necessidade de maiores estruturas de dados, cancelar a simulação quando da existência de erros nas informações fornecidas,etc.

Com o esquema adotado tornou-se possível estruturar blocos bem definidos como por exemplo a representação de uma pastilha de circuito integrado, e após testá-la, armazená-la em arquivos no disco, permitindo sua utilização quando necessário. A maneira de se utilizar os recursos do programa será melhor compreendida através de seu manual de uso que se apresenta como um dos elementos desta descrição. Apresenta-se, a seguir, as principais rotinas envolvidas na linguagem de entrada e uma descrição sucinta de como são utilizadas.

ROTINAS

READ - Esta rotina é utilizada para lêr as oitenta colunas de um cartão e armazená-las na área de memória reservada pelo bloco IREAD.

N72 - Esta rotina verifica, quando chamada, se o cartão lido e cujo conteúdo está na memória, possui continuação. Para isso, efetua o teste da posição do caráter em estudo; se a posição é de ordem menor que 72, nada há a executar mas se a posição é maior ou igual a 72, efetua-se um teste do caráter contido nesta posição. Havendo necessidade de um novo cartão, este será lido pela própria rotina; caso contrário, dará uma mensagem de falta de indicador de continuação. O controle da mensagem de erro é dado pelo indicador da posição dos caracteres. Esse ponteiro é indicado por N.

PACA - Esta rotina é utilizada para efetuar a compactação do nome do sinal. Dessa maneira é possível colocar 6 caracteres máximos, em apenas três palavras.

O nome do sinal é lido e colocado em uma matriz MAT de seis elementos pela rotina GETA; é compactado e colocado nos primeiros três elementos apenas. A matriz MAT está em COMMON.

CONVE - Esta rotina converte um número J de caracteres lidos, em um valor numérico. Os caracteres estão contidos numa matriz NAT de, no máximo, 6 elementos. A rotina efetua um teste para verificação se os caracteres lidos são números. Não sendo, uma mensagem de erro será emitida pela rotina.

Esta rotina possui dois argumentos:

"J" - número de caracteres a converter

"NAT" - matriz com os caracteres a converter

ERRO - Esta rotina permite sair com as condições de erro impressas. Para isso, usa variáveis contidas na área de COMMON, uma para indicar qual o número do erro e outra que deverá ser posicionada para indicar que houve erro.

GETA - Esta rotina é utilizada para lêr os caracteres contidos na área de memória carregada com os dados lidos de cartão. A rotina termina a leitura quando encontrar uma vírgula. Os caracteres lidos da área de memória são colocados na matriz MAT e retorna-se em J com o número de caracteres. A rotina aceita caracteres brancos e os ignora mas deteta erros quando um nome de sinal tiver mais que cinco caracteres (é o caso do presente programa) ou delimitador "C" no início do nome.

A variável N passa a apontar a primeira posição de caráter logo depois da vírgula.

Esta rotina possui dois argumentos:

"J" - número de caracteres lidos da memória

"NSAI" - indicador de sinal sempre igual a zero

OBS: Este indicador é utilizado pois esta rotina é também requisitada no "simulador de arquitetura", no qual necessita-se ter registradores, caso em que NSAI poderá ser um. Uma outra rotina, neste caso, será necessária. É a GETB que irá analisar o conteúdo de informação entre parênteses.

SIMBA - Esta rotina manipula a tabela de símbolos. Permite analisar os sinais contidos na tabela de símbolos e, se fôr necessário adicionar um sinal, ela o fará. Ao pesquisar um sinal na tabela, ela indicará o endereço do registro, se o sinal existir ou então colocará o sinal e indicará seu endereço. É ela que executa a segunda parte do algorítmo de "fan-in" e "fan-out". As informações necessárias à rotina são fornecidas através de argumentos e pela área de COMMON.

A tabela de símbolos é representada pela matriz MSIMB, a qual encontra-se também em COMMON.

Os argumentos desta rotina são os seguintes:

"NS" - número do bloco a considerar

"IPONT" - ponteiro que indica o endereço do sinal pesquisado ou colocado

"IE" - indica se é um sinal de entrada ("0") ou um sinal de saída ("1")

"IF" - indica ou o valor de "fan-in" ou o valor de "fan-out"

"INF" - indica se deve ("1") ou não deve("0") colocar sinal

Se após o retorno INF é negativo,isto indica que o sinal pesquisado não existe.

O nome do sinal a pesquisar encontra-se na matriz MAT em COMMON.

KONTR - Esta rotina lê o nome dos sinais de entrada de um bloco, a partir das informações existentes na memória. Utiliza a rotina SIMBA para colocar ou pesquisar sinais na tabela de símbolos.Os argumentos utilizados são os seguintes:

"I" - número de posições montadas na estrutura de dados

"NEAUX" - número de sinais de entrada

"KA" - última posição fixa da estrutura do bloco

"IND" - indica se a entrada é opcional ou não

"NS" - número do bloco a considerar

"MODO" - indica se entradas("0")ou saídas("1")

"IFIO" - indica o valor do "fan-in"ou do "fan-out"

A rotina também analisa as informações e envia mensagens de erros quando necessárias.

XDISC - É uma rotina para lêr ou gravar informações no disco. É utilizada tanto para gravar uma estrutura como também na segmentação de estruturas maiores.

Os argumentos necessários para esta rotina são os seguintes:

"IRC" - indica se leitura ("1") ou escrita ("2") no disco

"ISET" - indica o número de setores a lêr ou escrever

"IPRAM"- indica se deve lêr ou gravar a matriz da estrutura("1") ou a tabela de símbolos ("2").

A matriz da estrutura é representada pela área de COMMON reservada pela matriz MFUN.

Os arquivos em disco,nos quais são gravadas as duas informações, são respectivamente: #MFUN e #MSIMB.

LISTA - Esta rotina é utilizada para efetuar a listagem da estrutura de dados. Fornece esta lista em octal, decimal e asc. Os argumentos que necessita são apenas as informações do nome da matriz a listar e do ponteiro que indica até que ponto a listagem deve ser efetuada.

As rotinas que se seguem são chamadas pelo programa principal de acordo com o código representado pelos dois primeiros caracteres e analisam as demais informações contidas na memória. Estas rotinas não possuem argumentos, pois os dados necessários estão contidos na área de COMMON. Todas possuem esquemas de identificação de erros e respectivas mensagens, e são programas chamados como segmentos.

CROCO - Esta rotina é utilizada para processar as informações contidas nos cartões que possuem código CL ou CR nas duas primeiras colunas.

Os dados presentes no cartão que descreve este bloco são os seguintes:

"CR"ou"CL" - nome do sinal, período do sinal, largura do pulso,,

A rotina analisa as informações e as coloca na estrutura conforme mostra a figura 3.8.

GATO - Processa as informações contidas nos cartões, cujo código é GT. Os dados são os seguintes:

"GT" - nome do bloco, função do bloco, "fan-in", "fan-out", atraso, número de entradas,<entradas>,,

Os dados acima, uma vez analisados, são colocados na estrutura de acordo com o modelo especificado para o bloco. A figura 3.8 ilustra o esquema.

FLIPO - Permite a análise de blocos do tipo "FLIP-FLOP". Os dados a serem analisados por esta rotina são os que comparecem nos cartões com código FF, e lidos pelo programa principal.

"FF" - <nomes das saídas e "fan-out">, <atrasos>, <tempos e larguras de sinais>, borda, nível, função, <sinais de entrada e "fan-in"> ,,

A estrutura para este tipo de bloco é preenchida como mostra a figura 3.8.

GERAP - Esta rotina analisa as informações contida na memória e armazenadas a partir da leitura de cartões do tipo:

"GP" - nível lógico inicial, nome do sinal, periódico ou não, <valores> ,,

GRAVA - Esta rotina permite gravar blocos em arquivos no disco para posterior utilização. Quando se tem blocos a gravar, torna-se necessário dotá-los de uma estrutura flexível que permita sua fácil utilização. É o caso de representação de pastilhas, motivo pelo qual foi idealizada. Os sinais que se quer ter acesso ao bloco são informados logo após a colocação do nome do bloco. Os cartões que contêm essas informações são os que são lidos e que possuem como código os caracteres: GB.

Tratando-se de um conjunto de outros blocos, sua estrutura é bem complexa e se traduz por combinações de estruturas mostradas na figura 3.8.

Os dados necessários para a rotina são:

"GB" - nome do bloco, <sinais acessíveis>,,

Quando utilizada esta rotina, o bloco a ser gravado é colocado em arquivos e o seu nome na biblioteca do sistema que deverá conter também informações adicionais a respeito do bloco em questão.

<u>BLOCO</u> - Esta rotina permite lêr informações a respeito de blocos existentes no disco e montá-los na memória. Utiliza a mesma matriz fundamental da estrutura de dados e uma matriz especial para conter a tabela de símbolos do bloco lido do arquivo. As informações desta tabela são transferidas sob controle da rotina para a tabela de símbolos normal, à medida que estas são necessárias.

Os dados necessários para a rotina são:

"BL" - nome do bloco, <sinais a utilizar>,,

<u>DIREC</u> - Esta rotina é utilizada para listar as informações contidas na biblioteca do sistema a respeito dos blocos gravados. Fornece informações quanto ao nome do bloco gravado, dimensões do arquivo da estrutura de dados e da tabela de símbolos do bloco. É utilizada, também, quando se necessita efetuar a listagem da biblioteca, toda vez que se efetuar uma atualização na mesma.

CL ou CR

	num. bloco	C	S	R	período	duração pulso	nome do bloco	rasc

pont. (2) ou (3)

GP

	num. bloco	C	S	R	nível lógico	nome do bloco	num. de valores	Indíc period	Valor 1	Valor 2	.....	Indic. period.	Valor 1	Valor 2	...

pont. (5)

GT

num. bloco	C	S	R	atraso 0→1	atraso 1→0	num. sinais	função	nome do bloco	RASC 1	RASC 2	entradas

pont. (1)

função:
- 1 - NOR
- 2 - NAND
- 3 - NOT
- 4 - AND
- 5 - OR
- 6 - XOR

FF

						"clock"		"set"		"reset"				
num. bloco	C	S	R	nome da saída Q	nome da saída $\overline{Q}$	atraso 0→1	atraso 1→0	atraso 0→1	atraso 1→0	atraso 0→1	atraso 1→0	tempo "set-up"	tempo "hold"	largura "clock"

pont. (4)

largura "reset"	função	borda	nível	RASC 1a6	nome do "clock"	nome do "set"	nome do "reset"	DADO 1	DADO 2

função:
- 1- JK
- 2- D
- 3- MS

borda ou nível: 0→1 = 1; 1→0 = 0

Fig. 3.8 - Estrutura dos blocos

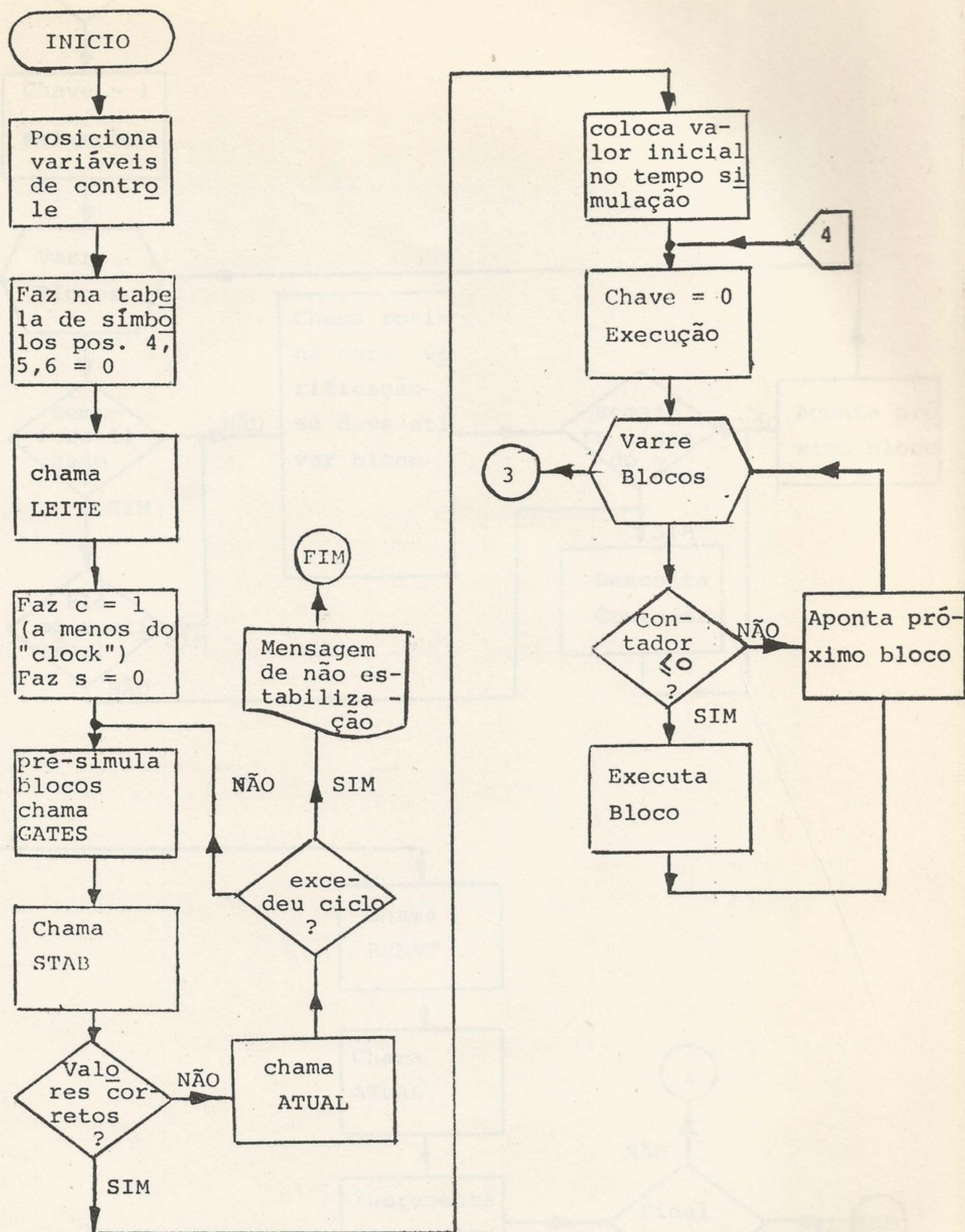



Fig. 3.9 - Estrutura do Simulador

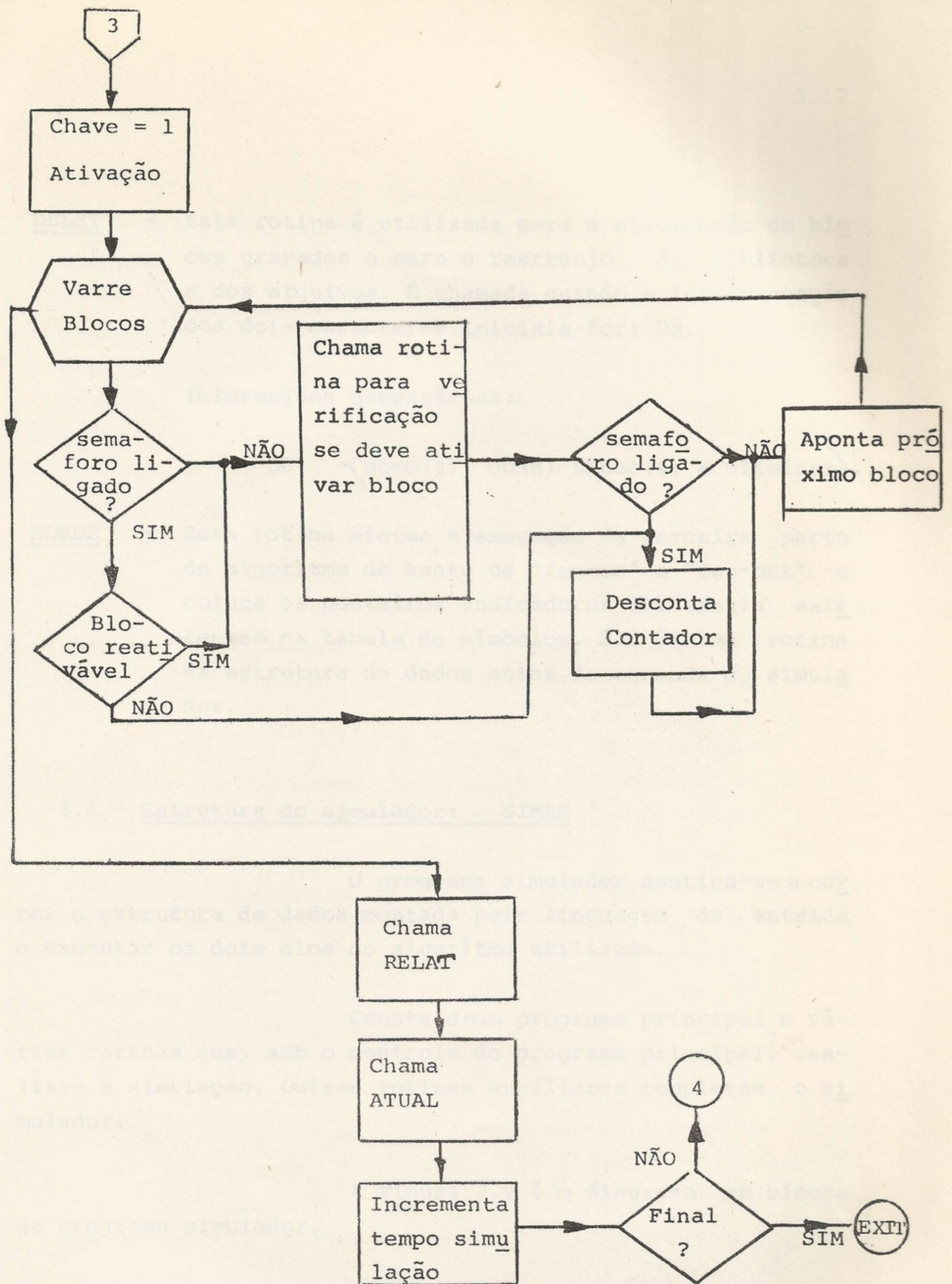



Fig. 3.9 - conclusão

DELET - Esta rotina é utilizada para a eliminação de blocos gravados e para o rearranjo da biblioteca e dos arquivos. É chamada quando a interpretação dos dois caracteres iniciais for: DB.

Informações necessárias:

"DB" -<nome(s) do(s) bloco(s) a eliminar>,,

MORDE - Esta rotina efetua a execução da terceira parte do algorítmo de teste de "fan-in" e "fan-out" e coloca os ponteiros indicadores dos sinais existentes na tabela de símbolos. É a última rotina da estrutura de dados antes da chamada do simulador.

3.6 - Estrutura do simulador: SIMAN

O programa simulador destina-se a correr a estrutura de dados montada pela linguagem de entrada e executar os dois elos do algorítmo utilizado.

Consta deum programa principal e várias rotinas que, sob o controle do programa principal, realizam a simulação. Outras rotinas auxiliares completam o simulador.

A figura 3.9 é o diagrama em blocos do programa simulador.

A maioria das rotinas opera com as informações contidas na área de COMMON; isto evita a presença de argumentos, facilitando a passagem de dados para as mesmas.

Inicialmente o programa principal se encarrega de colocar valores iniciais em determinadas variáveis que são necessárias ao processo.

Em seguida, efetua-se para todos os sinais presentes na tabela de símbolos, a colocação do valor zero nas palavras de número quatro, cinco e seis de cada registro com a exceção do primeiro, o qual contém os ponteiros para os sinais.

Após esse procedimento, o programa principal,através do chamado da rotina LEITE irá proceder à leitura de valores iniciais para os sinais e será,também, informado dos desejos do usuário quanto aos relatórios e duração da simulação.

Após a colocação de valores convenientes nas palavras contadora e semáforo de cada bloco,o simulador inicia uma pré-simulação dos blocos portas lógicas a fim de aceitar seus valores iniciais, ou para verificar se os mesmos estão coerentes com a função do bloco.Permite-se alguns ciclos até que se atinja a estabilidade dos mesmos. Após o tempo correspondente aos cilcos permitidos e não tendo-se chegado à estabilização, uma mensagem de alerta é impressa e a simulação termina.

Havendo estabilização, o simulador prosseguirá agora com a primeira fase do algorítmo ou seja o elo de execução. É nesta parte que o número associado a cada bloco terá sua participação, pois dele depende a chamada da rotina,característica do bloco em questão.

Após todos os blocos serem percorridos no elo de execução, o algorítmo inicia sua segunda fase com o início do elo de ativação.

Quando também esta fase terminar, a rotina de relatórios será chamada e deverá providenciar a documentação da situação dos sinais na forma escolhida pelo usuário.

Após o relatório, é efetuada a atualização dos sinais e do tempo de simulação, e o algorítmo retorna à sua primeira fase novamente.

As rotinas auxiliares, durante a simulação, manipulam dados existentes na estrutura principal e na tabela de símbolos. Essas rotinas são as seguintes:

<u>ATUAL</u> - Esta rotina permite efetuar a atualização dos valores dos sinais, através de um deslocamento de um "bit" em todos os registros nas palavras que contêm informações da simulação. Isto é feito para permitir que o "bit" mais significativo contenha o mesmo valor que o seguinte. Para isso o deslocamento que se efetua é aritmético. O "bit" mais significativo poderá ser substituído por um novo valor a ser fornecido pela simulação, ou ser mantido, quando não houver alteração durante a mesma.

Esta rotina apenas manipula a tabela de símbolos.

<u>RASAS</u> - Esta rotina é utilizada para colocar o valor simulado de um sinal no "bit" mais significativo da quarta palavra do registro ao qual o sinal pertence. Para isso, o endereço da quarta palavra do sinal é calculado pela rotina ENDER.

A utilização da rotina é efetuada colocando-se no acumulador A, o ponteiro para a posição onde está o sinal na estrutura, e no acumulador B o valor do sinal calculado. Em seguida, efetua-se o chamado da rotina RASAS. Como se nota, apenas a tabela de símbolos é manipulada pela rotina.

STAB - Esta rotina é utilizada apenas para efetuar o confronto entre os dois "bits" mais significativos da quarta palavra de todos os registros. A informação de retorno da rotina é fornecida através do acumulador B. Se para todos os registros a comparação entre os dois "bits" mais significativos , para cada quarta palavra, resultar coincidente,- então o acumulador B conterá o valor zero.

Caso contrário, um valor qualquer diferente de zero significará que pelo menos um sinal não estabilizou.

No programa principal, efetua-se o teste da estabilização dos sinais através do acumulador B.

Nesta rotina, a exemplo das duas anteriores, apenas a tabela de símbolos é pesquisada.

ENDER - Esta rotina calcula o endereço absoluto da quarta palavra do registro correspondente a um determinado sinal. Para tanto, no acumulador A é fornecida a posição do sinal na tabela de símbolos. O retorno da rotina é o endereço absoluto da quarta palavra do sinal, no acumulador B.

A rotina opera, também, como as anteriores, na tabela de símbolos.

CONTR - Esta rotina é utilizada para trazer valores dos sinais de entrada que compareceram na descrição de um determinado bloco. Para isso, deve-se fornecer à rotina, através do acumulador B,o número de sinais a considerar e, através do acumulador A, um ponteiro que indica a posição do primeiro sinal. Como retorno, obtém-se, no acumulador A, os valores atuais dos sinais ajustados à direita e no acumulador B os valores anteriores, também ajustados à direita. A rotina fornece, através de um ponteiro em COMMON, a próxima posição da esturura de dados a ser analisada, após os sinais de entrada.

Esta rotina utiliza a estrutura de dados e a tabela de símbolos como elementos de informação.

LEITE - É a rotina que permite efetuar a leitura de dados para o simulador.

Através dela, condições iniciais são estabelecidas para os sinais, tipos de relatórios são escolhidos, informados os tempos de impressão de sinais e a duração da simulação.

Esta rotina cria uma lista de sinais, cujos valores deverão ser impressos ou colocados em gráficos de forma de onda.

RELAT - É a rotina que decide o instante em que deve fornecer relatórios impressos dos sinais,ou efetuar o traçado da forma de onda dos mesmos.Para saber que sinais deve documentar, utiliza a lista de sinais criada pela rotina LEITE e presente na área

de COMMON. Através desta rotina, pode-se efetuar o cancelamento da simulação através da consulta às chaves de console do computador. Se a chave 1 estiver ligada, o processamento será cancelado.

ERRO - Esta rotina é utilizada para mensagens de erro do simulador.

XDISC - A rotina de disco aqui presente, é a mesma da linguagem de entrada, e utilizada para trazer à memória, arquivos existentes no disco, relativos à estrutura de dados e tabela de símbolos. Esses arquivos contém dados que foram armazenados pela linguagem de entrada, comandado pela presença de um cartão de controle que providencia o armazenamento da estrutura em disco. É utilizado pelo programa auxiliar SIMUL. Este programa só é chamado para colocar os dados da estrutura e tabela de símbolos que foram gravados no disco pelo usuário, na memória.

A rotina que trabalha com a segmentação, está presente no próprio programa principal, SIMAN.

As rotinas que se seguem são as que manipulam os blocos lógicos. São constituídas de duas partes que são processadas separadamente pelo elo de execução e pelo elo de ativação.

CROCK - Efetua a simulação de blocos CL ou CR.

Na fase correspondente ao elo de execução, a rotina gera sinal de acordo com a descrição fornecida pelo usuário e coloca o valor simulado na área do sinal na tabela de símbolos. Utiliza-se, para isso, das rotinas auxiliares.

Durante o <u>elo de ativação</u> a rotina nada executa, apenas retorna, o que vale dizer que o semáforo é sempre zero.

<u>GERPA</u> - Esta rotina manipula dados fornecidos por cartões do tipo GP.

Os sinais são gerados de acordo com a descrição fornecida pelo usuário e durante o <u>elo de execução</u> o valor gerado é colocado na área do sinal através do uso de rotinas auxiliares. Inúmeros são os controles executados durante esta fase, como por exemplo, se já foram gerados todos os valores, se o sinal necessita ser repetido etc. Durante o <u>elo de ativação</u>, a rotina apenas retorna, ou seja, o semáforo é sempre zero.

<u>GATES</u> - É a principal rotina do simulador pois é nela que o mesmo se apoia. Consta também de duas partes distintas e de um recurso adicional que permite à mesma ser utilizada pelo programa principal durante a pré-simulação. Nesta fase, a rotina apenas executa a função lógica da porta e coloca o valor simulado na variável que é a saída do bloco.

Pelo fato de ser uma rotina padrão do simulador em termos de utilização dos dois elos do algorítmo, apresenta-se o diagrama em blocos da mesma.

A figura 3.10 é o diagrama em blocos da rotina.

<u>ELO DE EXECUÇÃO</u>

Coloca, utilizando as rotinas auxiliares, o valor da saída simulada mantida no rascunho da estrutura na posição do sinal na tabela de símbolos. Co

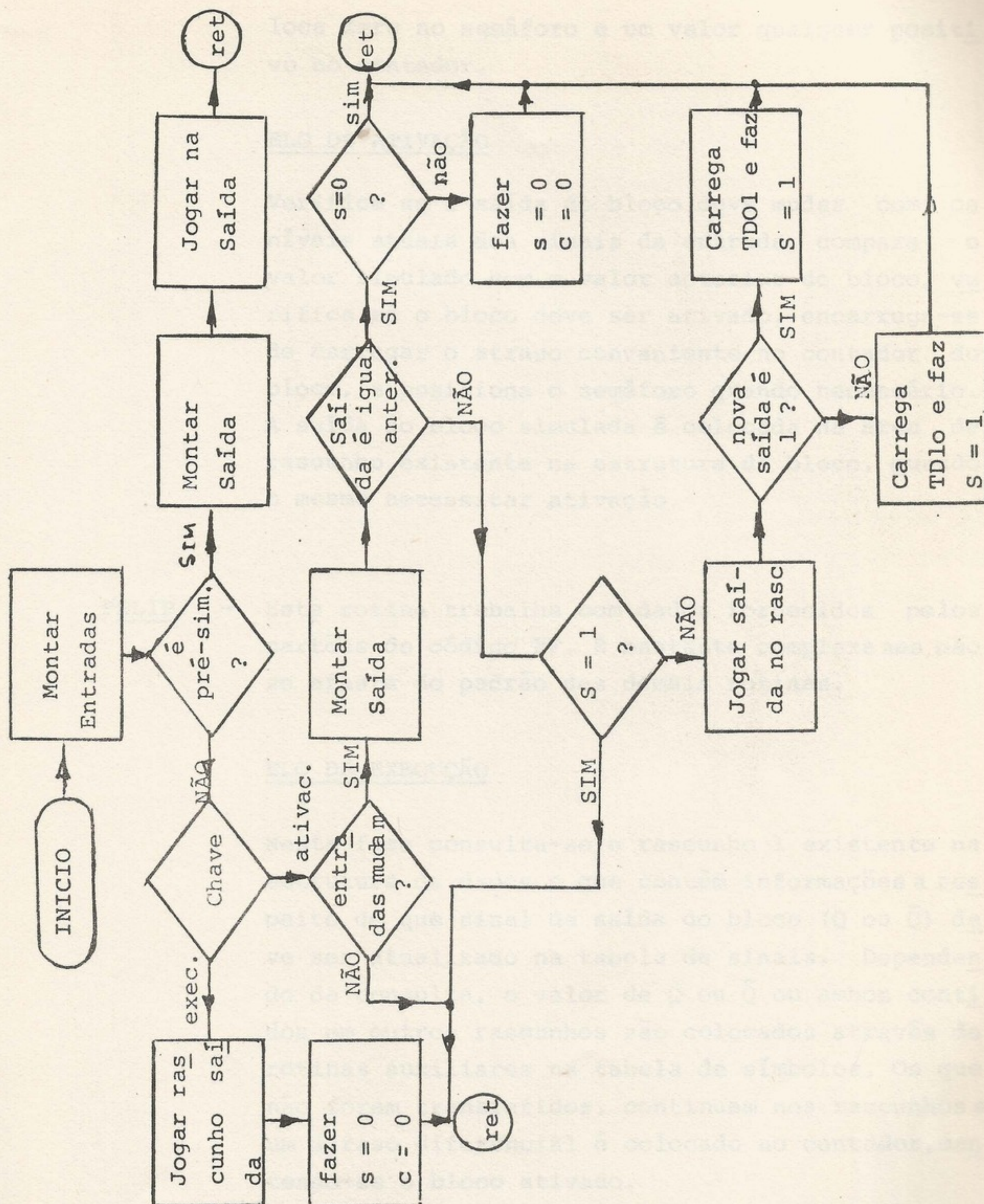



Fig. 3.10 -Estrutura típica de um bloco usando os dois elos

loca zero no semáforo e um valor qualquer positivo no contador.

ELO DE ATIVAÇÃO

Verifica se a saída do bloco deve mudar com os níveis atuais dos sinais de entrada, compara o valor simulado com o valor anterior do bloco, verifica se o bloco deve ser ativado, encarrega-se de carregar o atraso conveniente no contador do bloco, e posiciona o semáforo quando necessário. A saída do bloco simulada é colocada na área de rascunho existente na estrutura do bloco, quando o mesmo necessitar ativação.

FELIP - Esta rotina trabalha com dados fornecidos pelos cartões de código FF. É bastante complexa mas não se afasta do padrão das demais rotinas.

ELO DE EXECUÇÃO

Nesta fase consulta-se o rascunho 1 existente na estrutura de dados e que contém informações a respeito de que sinal de saída do bloco (Q ou $\bar{Q}$) deve ser atualizado na tabela de sinais. Dependendo da consulta, o valor de Q ou $\bar{Q}$ ou ambos contidos em outros rascunhos são colocados através de rotinas auxiliares na tabela de símbolos. Os que não forem transferidos, continuam nos rascunhos e um atraso diferencial é colocado no contador, mantendo-se o bloco ativado.

Se todos os valores forem atualizados, coloca-se zero no semáforo e um valor diferente de zero, mas positivo, no contador.

ELO DE ATIVAÇÃO

Durante esta fase, efetua-se testes nos sinais fornecidos pelo usuário para verificar se o bloco deve ser ativado ou não. Havendo necessidade de ativação, os valores de saída do bloco(Q, $\bar{Q}$), são simulados e colocados nos rascunhos, coloca-se no contador o valor do atraso conveniente e indica-se no rascunho 1 de informação, que sinal de saída deve ser atualizado em primeiro lugar. O semáforo é posicionado com o valor 1, indicativo de bloco ativado.

Durante esta fase, são efetuados testes com os sinais de dados e de duração dos sinais de controle. Esses testes são efetuados apenas quando o usuário o desejar.

4. Manual de utilização do programa de simulação em nível de portas.

4.1 - Entrada de dados e preparação dos cartões

.A entrada de informações para o programa é efetuada através de cartões perfurados. Pode-se a qualquer momento, se assim o usuário o desejar, efetivar a entrada de dados através de fita magnética ou fita de papel, bastando que se efetue a substituição do dispositivo de entrada para o programa. (O sistema HP é bastante flexível nesse aspecto.)

.Os cartões devem descrever a interligação do sistema pela descrição de cada um dos seus blocos funcionais.

.Dependendo do bloco funcional, que se deseja descrever, varia o formato do cartão de dados. A única informação fixa que esses cartões possuem, dizem respeito às suas primeiras colunas nas quais existe o código do bloco a ser utilizado.

.Para a codificação de dados podem ser utilizadas as colunas de 1 a 71 do cartão. Se a informação deve continuar em outro cartão, um (*) deve ser perfurado na coluna 72 do cartão.

.Dentro de um cartão a codificação é de formato livre, sendo os campos separados um dos outros por meio de ",". O fim da informação é caracterizado por",," e pela análise de outras informações já obtidas do cartão.

.Quando falta uma informação (que seja opcional) deve-se colocar no cartão apenas as duas vírgulas seguidas ",," indicando o fato.

. Caracteres brancos são ignorados.

.Os nomes de sinais devem conter no máximo cinco caracteres quaisquer, a menos de "(".

4.2 - Códigos de identificação do cartão

. Nas duas primeiras colunas do cartão deve comparecer um dos seguintes códigos:

BL - Montar bloco na memória
CL - gerador de "clock" normal
CR - gerador de "clock" reverso
DB - apagar bloco existente na biblioteca do sistema
DI - gravar em disco
FF - bloco "FLIP-FLOP"
FI - fim de descrição da estrutura
GB - guardar bloco na biblioteca do sistema
GP - gerador de sinais
GT - bloco porta lógica
LB - listar blocos existentes na biblioteca do sistema
LI - listar a estrutura de dados montada
NB - novo bloco a gravar
TE - término de uso da linguagem de entrada
** - comentário

4.3 - Formato genérico de um cartão de dados

. Cada bloco possui um formato diferente e variável de cartão de entrada que contém as informações acerca de sua estrutura. Assim sendo, para se obter o formato exato e a maneira correta de preenchimento dos cartões de entrada, deve-se consultar sempre a descrição do cartão de cada bloco funcional, em confronto com os exemplos apresentados.

. O formato genérico do cartão é o seguinte:

```
CB, nome do bloco,   atrasos  ,"fan-in", "fan-out", função,
<entradas> ,,
```

onde,

CB - é o código que identifica o bloco funcional

nome do bloco - é o nome identificador do próprio bloco e deverá conter o estado atual e os valores anteriores da simulação do bloco. Representa, na verdade, o nome do sinal que sai do bloco.

atrasos - correspodem aos atrasos que o bloco possui e que são necessários durante a simulação. Geralmente são em número de dois: o atraso que corresponde ao tempo necessário para a saída passar do estado "0" para o estado "1" (TD01) e o atraso que corresponde ao tempo necessário para a saída passar do estado "1" para o estado "0" (TD10). Sua unidade de tempo é em nanosegundos e o valor unitário de um atraso é o tempo unitário ou passo da simulação.

"fan-in" - é um número que indica o valor de carga normalizada para cada entrada do bloco

"fan-out" - é um número que indica o valor de carga normalizada para a saída do bloco

função - indica qual a função a ser executada pelo bloco funcional, quando assim se fizer necessário, (por exemplo: função AND, OR, NOR etc.).

entradas - é a lista dos nomes dos sinais de entradas para o bloco. Quando necessário deverá ser acompanhada das informações de "fan-in" ou de "fan-out".

4.4 - Formato de cartões e blocos funcionais permitidos

4.4.1 - Blocos funcionais

. porta lógica ("gates"): GT

O cartão de entrada deve ser codificado da seguinte forma:

GT, nome do bloco, função, "fan-in", "fan-out",
TD01, TD10, número de entradas, <entradas>,,

onde:

funções permitidas para a porta lógica são:
AND, NOR, NAND, NOT, OR, XOR

nº de entradas- é o número de sinais que alimentam o bloco, ou seja, o número de entradas ocupadas.

Exemplos:

GT, RB1, AND, 1, 1, 5, 5, 2, D1, STROB ,,
GT, FLOS, NOT, 1, 10, 1, 2, 1, FLIS ,,
GT, QB, NOR, 1, 16, 3, 1, 2, FLIS, FLOS ,,

. bloco "FLIP-FLOP": (FF)

As informações que devem ser fornecidas através deste cartão são as seguintes:

FF, saída Q,"fan-out", saída $\bar{Q}$, "fan-out", TD01, TD10, TD01,
*1 *1 *1
TD10, TD01, TD10, t"set-up", t"hold", Larg"clock", Larg"set",
*2 *3 *3
Larg"reset", borda, nível, função, nome do "clock", "fan-in",
nome do "set", "fan-in", nome do "reset", "fan-in", Dado 1,
"fan-in", Dado 2, "fan-in" ,,

onde:

*1	dizem respeito ao sinal de "clock"
*2	dizem respeito ao sinal de "set"
*3	dizem respeito ao sinal de "reset"
t"set-up"	é o tempo de "set-up" para o dado
t"hold"	é o tempo de "hold" para o dado
Larg"clock",	larg"set", larg"reset" - correspondem às larguras necessárias aos sinais de "clock","set", "reset"
borda	especifica que o bloco é sensível à borda do "clock", ou seja, ou de 0→1 (01) ou de 1→0 (10)
nível	indica que os sinais atuam quando em nível lógico "0" (10)ou nível lógico "1" (01)

função	o bloco "FLIP-FLOP" pode executar as funções de bloco JK, D ou Ms.

dado 1	no caso do bloco ter função de JK, é o J; no caso de função D, é o próprio dado.

dado 2	no caso do bloco ter função JK, é o K; no caso de função D, deve-se colocar ",".

Observações:

1. As informações desde t"set-up" até larg"reset" são opcionais e não existindo dados deve-se colocar a condição de opcional.
2. Os sinais de "clock", "set" e "reset" também são opcionais. No caso de um deles não aparecer, o correspondente "fan-in" não deverá constar no cartão.
3. As larguras e tempos de "set-up" e "hold" não devem ser maiores que 32 unidades de tempo.

Exemplos:

"master-slave" completo:

FF, Q, 1, QB, 1, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 01, 01, MS, CL 1, SET, 1, RESET, 1, J, 1, K, 1 ,,

"JK" com "clock" e dados e sem testes de tempos e larguras:

FF, Q, 1,QB, 2, 1, 1, 1, 1, 2, 2,,,,,,01, 01, JK, CL, 1,,,J, 1, K, 1 ,,

.Gravação de blocos lógicos definidos pelo usuário: (GB)

O uso de blocos, como o "FLIP-FLOP" definido acima, é bastante deficiente sob o ponto de vista de utilização. O presente programa permite ao usuário a definição de blocos "FLIP-FLOP" em nível de portas lógicas, seu teste, gravação e posterior utilização como se fôsse uma pastilha de circuito integrado. Para tanto, uma vez efetuada a-

definição do bloco a se utilizar, em nível de portas lógicas, basta que o usuário efetue a escolha dos sinais que deseja ter acesso e os coloque como informações no cartão que efetua a gravação.

Cada bloco terá um nome que deverá constar da descrição do cartão. Este nome será pesquisado na biblioteca do sistema e, não havendo duplicação, será arquivado.

O cartão que permite gravar blocos tem o seguinte formato:

GB, nome do bloco, <nome dos sinais acessíveis>,,

A sequência dos sinais é importante e devem ser os mesmos existentes na descrição da estrutura do bloco que se deseja gravar.

Exemplos:

1. Após os cartões em nível de porta lógica e que descrevem um "DUAL JK - NEGATIVE EDGE TRIGGERED FLIP-FLOP" que constitui a pastilha de nome 74S113, deve-se colocar o cartão de gravação:

GB, S113, PRESE, CLEAR, CLOCK, J, K, Q, QB ,,

onde: PRESE, CLEAR, CLOCK, J, K, Q, QB, são os sinais acessíveis.

2. Após a descrição de um "DUAL D TYPE EDGE TRIGGERED FLIP-FLOP", que constitue a pastilha 74S74, deve-se colocar o cartão:

GB, S7474, PRESS, CLEAR, CLOCK, DADO, Q, QB ,,

onde: PRESS, CLEAR, CLOCK, DADO, Q e QB são os sinais acessíveis.

.Montagem de blocos definidos pelo usuário: BL

Após a gravação de blocos especificados pelo usuário, como mostrado no item anterior, o mesmo poderá ser utilizado quando desejado, bastando apenas que seja

chamado através de um cartão do tipo BL. O nome do bloco deverá ser um dos presentes na biblioteca do sistema e os sinais que se quer utilizar, terão o mesmo significado que na gravação, mas o nome poderá ser aquele que o usuário necessitar na montagem do seu sistema.

O usuário poderá utilizar o mesmo bloco, o número de vezes que julgar necessário.

O cartão que permite a montagem dos blocos tem as seguintes informações:

BL, nome do bloco, <sinais a utilizar> ,,

Exemplos:

A chamada para montagem da pastilha S7474, do item anterior, será:

BL, S7474, PRESE, LIMPA, RELOL, DADO, FLIP, FLOP ,,

.Eliminação de blocos da biblioteca: DB

Os cartões que possuem nas duas primeiras colunas o código DB, permitem a eliminação de blocos arquivados e cujos nomes encontram-se presentes na biblioteca do sistema. Após a eliminação efetua-se uma compactação dos arquivos e atualização da biblioteca.

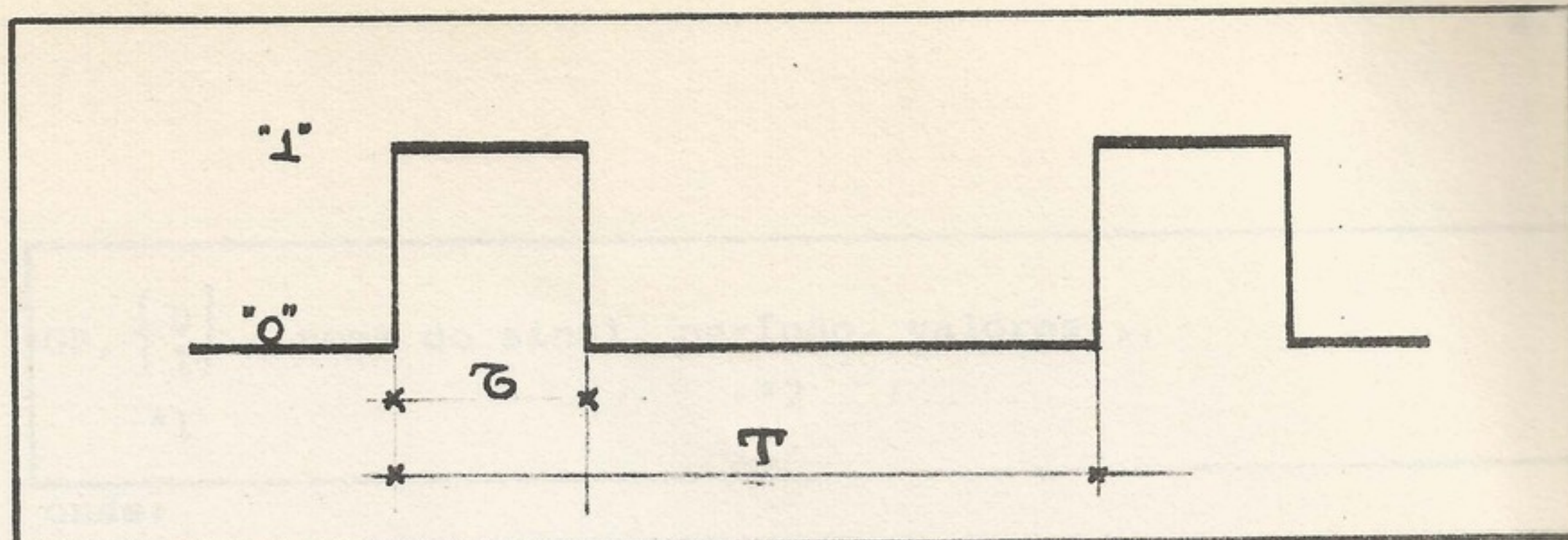
O formato do cartão é o seguinte:

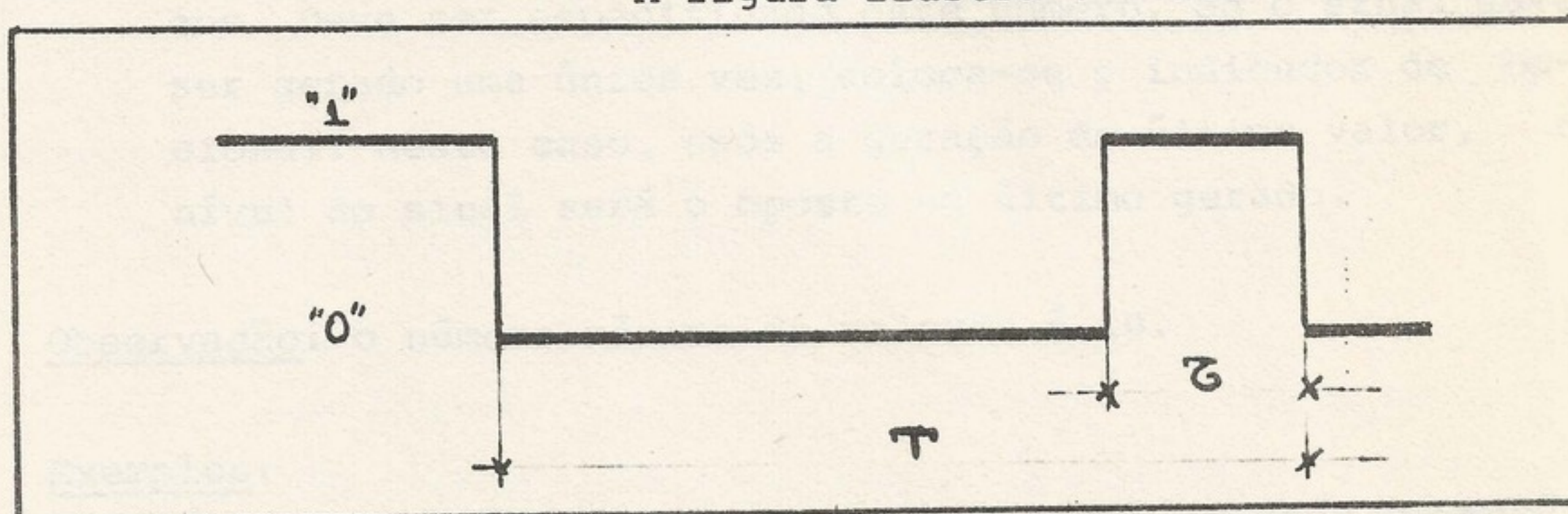
DB, <nomes dos blocos> ,,

Exemplos:

DB, S113, S7474 ,,

.Gerador de "clock" normal e reverso: CL, CR

Os cartões do tipo CL e CR permitem gerar sinais de "clock" com período e largura definidos. O "clock" normal é um sinal que inicia-se no nível lógico "0" e vai para o nível "1", como mostra a figura:

O "clock reverso é idêntico; apenas que no início o sinal sai do nível "1" e vem para "0".

A figura ilustra o "clock" reverso:

Os cartões que descrevem os geradores de "clock" tem as seguintes informações:

{CL} , nome do sinal, período, duração do pulso ,,   {CR}

Exemplos:

CL, CLOCK, 100, 30 ,,

CR, CROCO, 100, 30 ,,

.Gerador de sinais: GP

O gerador de sinais permite a formação de um sinal desejado para testes pois sua composição é deixada ao critério do próprio usuário. Como no caso do gerador de "clock", aqui tembém torna-se necessário especificar qual o nível lógico de partida.

Na especificação de um sinal, a duração do intervalo no qual o sinal deve permanecer num determinado nível lógico é denominado de valor.

As informações necessárias são as seguintes:

```
GP, {0} , nome do sinal, período, valores ,,
    {1}
    *1                             *2
```

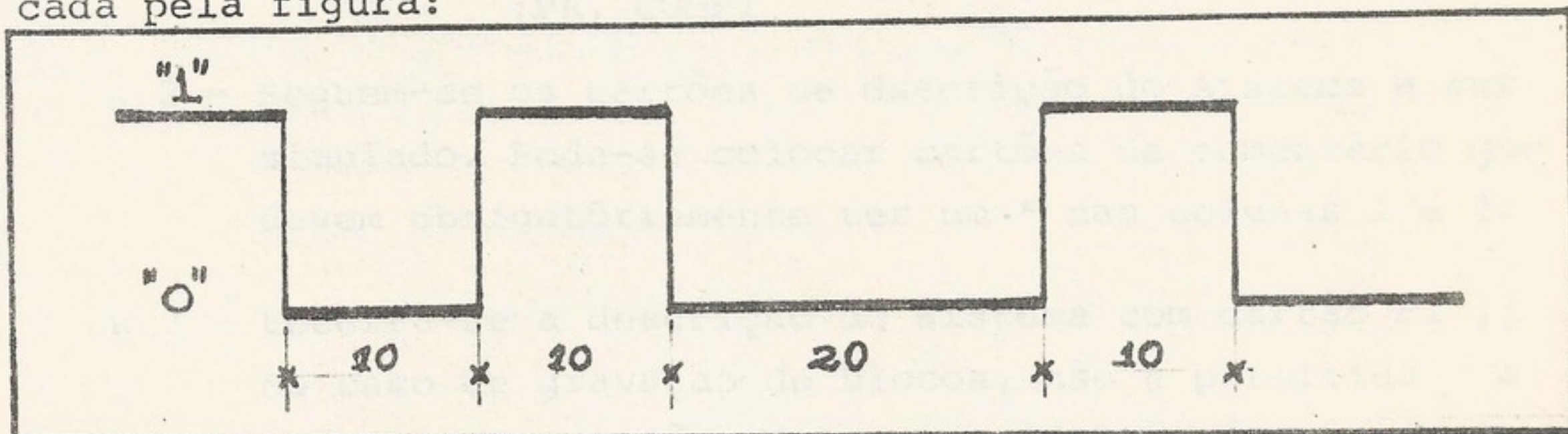
onde:

*1 indica o nível lógico de início. Se "0", o sinal começa no nível "1" e vem para "0".

*2 o sinal gerado pode ser repetido um certo número de vezes. Deve ser especificado este número. Se o sinal deve ser gerado uma única vez, coloca-se o indicador de opcional. Neste caso, após a geração do último valor, o nível do sinal será o oposto ao último gerado.

<u>Observação</u>: o número máximo de valores é 10.

<u>Exemplos</u>:

A geração de um sinal com a forma de onda especificada pela figura:

será obtida com o cartão:

GP, 0 , SAIDA ,, 10, 10, 20, 10 ,,

4.4.2 - <u>Cartões de controle para a linguagem de entrada</u>

O programa que efetua a leitura dos cartões de entrada, que descrevem a estrutura de dados a ser montada para o simulador, é gerenciado através dos seguintes tipos de cartões:

COLUNA 1	
NB ,,	(Novo Bloco)
LB ,,	(Listar Blocos)
DI ,,	(Discos)
FI ,,	(final)
LI ,,	(Lista)
TE ,,	(Termina)
**	(Comentário)

A utilização de cada cartão é assim descrita:

a - Sequência de entrada para a linguagem de entrada

a.1 - Os cartões DI ,, e LI ,, devem, se desejada a sua utilização, vir logo a seguir do cartão de chamada do programa que constitue a linguagem de entrada.

:PR, COMPI

a.2 - Seguem-se os cartões de descrição do sistema a ser simulado. Pode-se colocar cartões de comentário que devem obrigatòriamente ter um * nas colunas 1 e 2.

a.3 - Encerra-se a descrição do sistema com cartão FI ,,. No caso de gravação de blocos, não é permitida a presença do cartão FI ,, e sua função é substituida pelo cartão de gravação (GB) o qual encerra a descrição do bloco e o armazena no disco. Entre um bloco e outro deve comparecer o cartão NB ,,.

a.4 - Vem a seguir os cartões ou de término da linguagem de entrada (TE ,,) ou de chamada do simulador (SI,,).

a.5 - O cartão LB ,, permite listar a biblioteca de blocos gravados.

b)- Função de cada cartão

LI,, permite, quando presente, listar a matriz fundamental estruturada para o sistema em questão. Sua utilidade é na depuração de programas.

DI,, indica quando presente, que a estrutura montada a partir dos cartões deve, se não houver erros, ser arquivada no disco para uso imediato e futuro.

FI,, sua presença é obrigatória. Indica para a linguagem de entrada a finalização de descrição dos dados do sistema.

TE,, quando presente encerra o processamento. (Só é utilisado para a linguagem de entrada).

SI,, é utilizado para, através da linguagem de entrada, efetuar a chamada do simulador.

Não é necessário explicitar se os dados do sistema estão guardados no disco ou não. A própria linguagem se encarrega de verificar onde se encontram os dados para simulação.

NB,, inicia os ponteiros da linguagem de entrada para receber novo bloco.

4.4.3 - Cartões de controle para o programa simulador

O programa que efetua a simulação do sistema possui também cartões de controle e estes são os seguintes:

TIPO 1 : Inicialização de sinais
TIPO 2 : Informações Gerais
Cartão de finalização

a - Sequência de entrada dos cartões acima

A ordem é qualquer sendo que o cartão de finalização deve ser o último.

b - Função de cada cartão:

b.1 - Cartões do TIPO 1: estabelecimento de condições iniciais para os sinais.

Estes cartões apresentam como característica o fato de possuirem nas duas primeiras colunas o código de condição inicial (CI).

Os cartões podem conter dados até a coluna 71. Os nomes de sinais não podem ser truncados e continuados em outros cartões. Deve-se colocá-los por extenso em novo cartão, se necessário, o qual deve ter as iniciais correspondentes ao tipo de cartão considerado.

```
COLUNA 1
↓
CI ȻȻ XXXXXX = Valor, YYYYYY = Valor, .......
```

onde:

XXXXXX ou YYYYYY é nome do sinal a ter condição inicial seguido imediatamente do sinal = e de um número cujo valor é ou "0" ou "1".

.Não é permitido brancos na descrição.

.Não existe cartão de continuação e o usuário deve começar a descrição de outros cartões, se necessário, sempre com o código CIȻȻ seguindo-se os nomes dos sinais com seus valores.

b.2 - Cartões do TIPO 2

b.2.1 - Solicitação de impressão de sinais

Os cartões podem conter dados até a coluna 71. Os sinais não podem ser truncados e continuados em outros cartões. Deve-se colocá-los por extenso em novo cartão, se necessário, o qual deve ter as iniciais correspondentes ao tipo de cartão considerado.

Os sinais que se deseja imprimir são fornecidos através do cartão:

```
COLUNA 1
↓
RS ȻȻ xxxxxx, yyyyyy, zzzzzz, ..., wwwwww
```

onde:

XXXXXX, YYYYYY, ZZZZZZ, WWWWWW são os nomes dos sinais a imprimir.

Não existe cartão de continuação e o usuário deve colocar tantos cartões RSØØ quanto necessários.

Observação:

. Número máximo de sinais a imprimir = 40

. Havendo cartões para saida de formas de onda vertical, os dados solicitados para impressão serão ignorados.

b.2.2 - Solicitação de formas de onda

Os cartões podem conter dados até a coluna 71. Os nomes de sinais não podem ser truncados e continuados em outros cartões. Deve-se colocá-los por extenso em novo cartão, se necessário, o qual deve ter as iniciais correspondentes ao tipo de cartão considerado.

Pode-se solicitar a impressão da forma de onda de sinais através dos cartões:

b.2.2.1 - Forma de onda horizontal

```
COLUNA 1
↓
PHØØ, XXXXXX, YYYYYY, ZZZZZZ, ... , WWWWWW
```

onde:

XXXXXX, YYYYYY, ZZZZZZ, WWWWWW são nomes dos sinais cujos gráficos são desejados.

.Não deve haver brandos na descrição.

.Não existe cartão de continuação e o usuário deve colocar tantos cartões PHØØ quanto necessários.

Observação;

.Número máximo de sinais para gráfico = 40.

b.2.2.2 - Forma de onda vertical

O formato é o mesmo que o anterior, apenas com as seguintes observações:

1º) O código de saida gráfica é PV.

2º) O número máximo de sinais para gráfico é 11.

b.2.3 - Intervalo de impressão de sinais

Pode-se solicitar que o intervalo de impressão seja estabelecido a critério do usuário. Isso é efetuado através do cartão:

```
COLUNA 1
↓
INTERVALO ƀ DE ƀ TEMPO ƀƀ ØØØØØØ
```

onde:

ØØØØØØ é o valor do intervalo em octal.

A ausência deste cartão é interpretada pelo simulador como sendo o intervalo de tempo unitário.

Este cartão tem significado apenas no caso de se ter cartões RS na descrição.

b.2.4 - Tempo máximo de simulação

A duração da simulação é determinada pelo cartão;

```
COLUNA 1
↓
TEMPO ƀ FINAL ƀ ØØØØØØ
```

onde:

ØØØØØØ é o valor do instante final de simulação, em octal.

A ausência deste cartão é interpretada como simulação durante 10.000 intervalos unitários.

O usuário pode ainda, através de manuseio da chave Ø do painel, interromper o processamento quando o desejar.

b.3 - Cartão de finalização

Este cartão é obrigatório e encerra a leitura de dados e informações para o simulador. Não havendo dados ou informações gerais o cartão de finalização será o único a ser lido pelo simulador.

O formato deste cartão é o seguinte:

```
COLUNA 1
↓
FIM
```

NOTA IMPORTANTE: Se os dados da estrutura foram arquivados no disco através do cartão DI,, ou então automáticamente pela imagem de entrada e não foram destruidos, estes dados poderão ser simulados diretamente utilizando-se o seguinte cartão:

:PR, SIMUL

Seguem-se os cartões de uso do simulador.

4.5 - Controle de erros

4.5.1 - Erros detetados pela linguagem de entrada

ROTINA DE CONVERSÃO

20 - conversão de dado alfabético

ROTINA DE BUSCA DE CARACTERES

22 - dentro dos parênteses, encontrada uma vírgula logo após o "("

30 - primeiro caracter de um nome é "("

31 - nome com mais de cinco caracteres

40 - logo após ")" não tem vírgula

41 - dentro dos parênteses, encontrada mais que uma vírgula

ROTINA DE CONTROLE DE SINAIS

10 - não encontrado nome de controle a analisar no cartão de dado

11 - o número de bits é diferente de 1

2ØØ - encontrado nome de sinal com "("

1ØØ - erro ou falta de nome de um sinal de um FF

1Ø1 - não se especificou "fan-in" - "fan-out" de um sinal num FF

1Ø2 - um sinal foi utilizado como registrador

ROTINA DE CONTROLE DE DIMENSÃO DE UM CARTÃO

5ØØ - não se encontrou um * na coluna 72 para indicar continuação

ROTINA PARA PESQUISAR CARTÕES DO TIPO CL OU CR

70 - falta o nome do sinal de "clock" ou colocou-se parênteses no nome

71 - não colocado o valor de "T" ou "TAU"

ROTINA PARA PESQUISA CARTÕES DO TIPO GP

9Ø1 - não especificado o "0" ou "1" corretamente

9Ø3 - não especificado o nome do sinal do Gerador de palavras

ROTINA PARA PESQUISAR CARTÕES DO TIPO FF

51 - nome do sinal Q ou QB não especificado ou então não especificado "fan-in"/"fan-out" ou então não especificado o tipo do FF

52 - encontrado um sinal "(" ou ")" no nome do sinal Q ou QB

53 - não especificado TD01 ou TD10 do: clock ou set ou reset

54 - erro na especificação da borda ou nível

55 - função não existente

ROTINA PARA PESQUISAR CARTÕES DO TIPO GT

501 - não especificado o nome do bloco "GATE"ou sua função, ou ainda não especificado "fan-in" / "fan-out" ou atrasos do"GATE"

502 - no nome do sinal existe delimitador "("ou ")"

503 - não existe função especificada

ROTINA PARA PESQUISAR CARTÕES DO TIPO GB

700 - erro no nome do bloco

701 - não especificado o nome do bloco

702 - nome do bloco já existe

703 - erro no nome de um sinal

704 - foi gravado parte do bloco no disco (o tamanho do bloco ultrapassa a dimensão padrão)

705 - não existe sinal especificado no cartão GB

706 - tentativa de gravação do bloco com erro de entrada

ROTINA PARA PESQUISAR CARTÕES DO TIPO BL

601 - erro no nome do bloco

602 - não especificado o nome do bloco

603 - erro em nome de sinal

604 - não há correspondência entre parâmetro especificado no BL e parâmetro do bloco gravado

605 - faltam sinais na especificação do bloco

606 - excedeu o número de sinais permitidos para o bloco

ROTINA PARA PESQUISAR CARTÕES DO TIPO DB

800 - erro no nome do bloco

PROGRAMA PRINCIPAL DA LINGUAGEM DE ENTRADA

42 - o código existente no cartão não é de controle ou indicativo de bloco

4.5.2 - Erros detetados pelo simulador

91 ERRO NA ROTINA DE DISCO.Não encontrados arquivos para armazenamento de dados

01 sinal na lista de entrada para forma de onde ou imprimir ou dar condições iniciais, inexistentes

5. Exemplos de uso do programa de simulação

5.1 -Exemplos de utilização da linguagem de entrada e do simulador.

Apresentam-se nesta secção diversos exemplos de utilização, que podem auxiliar os que desejarem se familiarizar com o programa de simulação de sistemas digitais e servir-se dele.

A maior parte dos exemplos é constituida de portas lógicas e define uma estrutura que é gravada em disco para posterior utilização.

Apresenta-se ainda o exemplo do relógio central de um computador em desenvolvimento no Laboratório de Sistemas Digitais, onde são utilizados vários blocos representativos de pastilhas de circuitos integrados que constam da biblioteca do sistema e cujo tipo pode ser conhecido, atráves da consulta aos circuitos apresentados.

Os desenhos inclusos servem como referência para melhor compreensão dos blocos codificados.

5.1.1 Circuito (fig. 5.1) e cartões de descrição do: 74278

"four bit cascadable priority register"

74278 — 4 BITS CASCADABLE PRIORITY REGISTER
PØ
STROB
D4
D3
D2
D1
RB1
SB1
DB1
QB1
Q1
RB2
SB2
DB2
QB2
Q2
RB3
SB3
DB3
QB3
Q3
RB4
SB4
DB4
QB4
Q4
Y2
Y3
Y4
P1

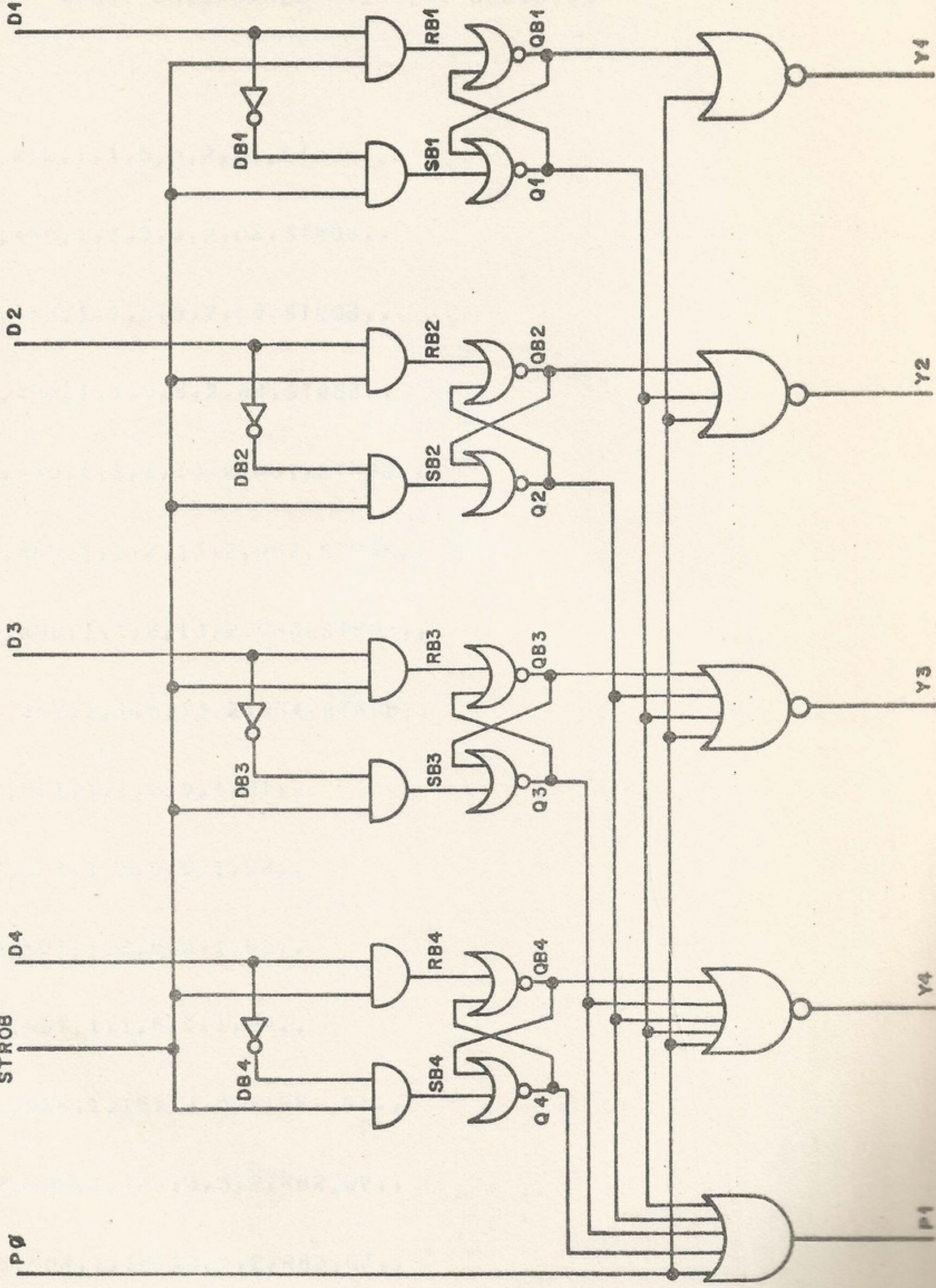



FIG. 5.1

```
**

**74278 - 4-BIT CASCADABLE PRIORITY REGISTER

**

GT,RB1,AND,1,1,5,5,2,D1,STROB,,

GT,RB2,AND,1,1,5,5,2,D2,STROB,,

GT,RB3,AND,1,1,5,5,2,D3,STROB,,

GT,RB4,AND,1,1,5,5,2,D4,STROB,,

GT,SB1,AND,1,1,2,13,2,DB1,STROB,,

GT,SB2,AND,1,1,2,13,2,DB2,STROB,,

GT,SB3,AND,1,1,2,13,2,DB3,STROB,,

GT,SB4,AND,1,1,2,13,2,DB4,STROB,,

GT,DB1,NOT,1,1,6,5,1,D1,,

GT,DB2,NOT,1,1,6,5,1,D2,,

GT,DB3,NOT,1,2,6,5,1,D3,,

GT,DB4,NOT,1,1,6,5,1,D4,,

GT,QB1,NOR,1,10,13,5,2,RB1,Q1,,

GT,QB2,NOR,1,10,13,5,2,RB2,Q2,,

GT,QB3,NOR,1,10,13,5,2,RB3,Q3,,

GT,QB4,NOR,1,10,13,5,2,RB4,Q4,,

GT,Q1,NOR,1,10,5,2,2,SB1,QB1,,
```

```
GT,Q2,NOR,1,1Ø,5,2,2,SB2,QB2,,

GT,Q3,NOR,1,1Ø,5,2,2,SB3,QB3,,

GT,Q4,NOR,1,1Ø,5,2,2,SB4,QB4,,

GT,Y1,NOR,1,1Ø,2Ø,16,2,QB1,PØ,,

GT,Y2,NOR,1,1Ø,2Ø,16,3,QB2,Q1,PØ,,

GT,Y3,NOR,1,1Ø,2Ø,16,4,QB3,Q1,Q2,PØ,,

GT,Y4,NOR,1,1Ø,2Ø,16,5,QB4,Q1,Q2,Q3,PØ,,

GT,P1,OR,1,1Ø,23,3Ø,5,Q1,Q2,Q3,Q4,PØ,,

**

**

**  VAI ARMAZENAR BLOCO

**

GB,B278,STROB,PØ,D1,D2,D3,D4,P1,Y1,Y2,Y3,Y4,,

ND,,
```

5.1.2 Circuito (fig. 5.2) e cartões de descrição do: 74S113 "dual JK negative edge triggered flip-flop"

74S113 - DUAL J-K - NEGATIVE EDGE TRIGGERED FLIP-FLOP
Q
QB
FLI
FLO
FLIS
FLOS
CLEAR
J
IN1
IN2
PRESE
K
CLOCK

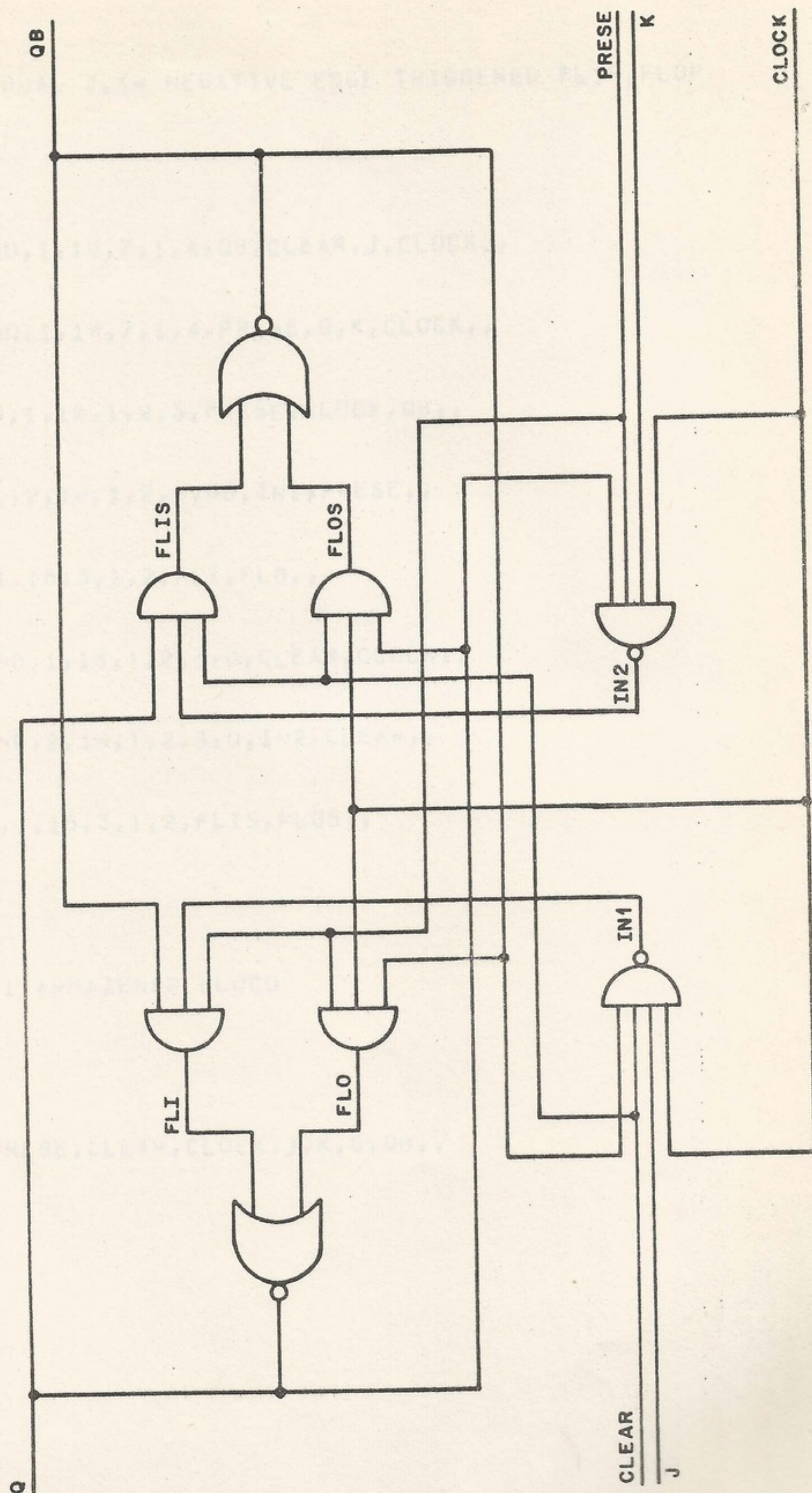
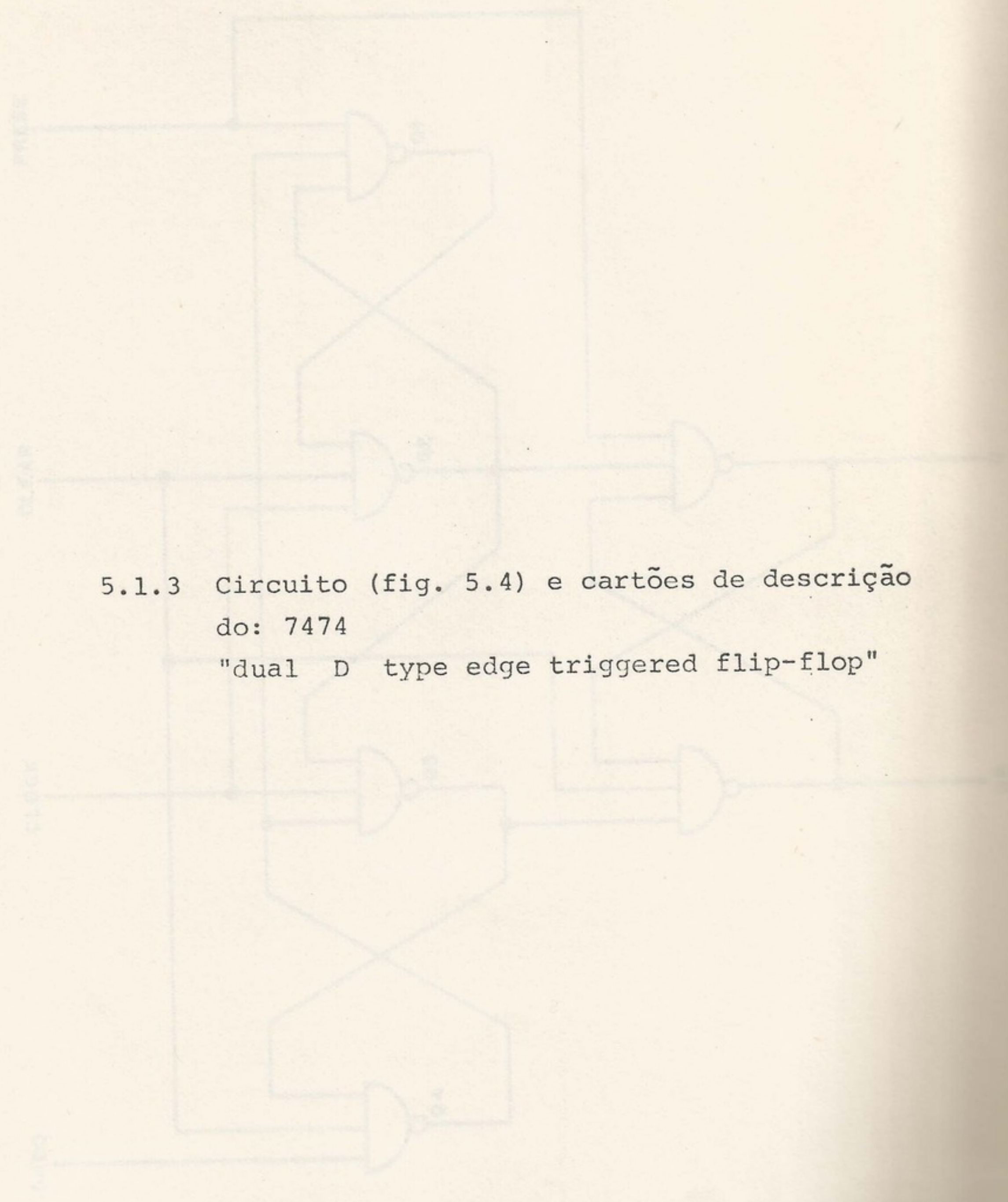



FIG. 5.2

```
**

**74S113- DUAL J.K- NEGATIVE EDGE TRIGGERED FLIP.FLOP

**

GT,IN1,NAND,1,10,7,1,4,QB,CLEAR,J,CLOCK,,

GT,IN2,NAND,1,10,7,1,4,PRESE,Q,K,CLOCK,,

GT,FLO,AND,1,10,1,2,3,PRESE,CLOCK,QB,,

GT,FLI,AND,2,10,1,2,3,QB,IN1,PRESE,,

GT,Q,NOR,1,16,3,1,2,FLI,FLO,,

GT,FLOS,AND,1,10,1,2,3,Q,CLEAR,CLOCK,,

GT,FLIS,AND,2,10,1,2,3,Q,IN2,CLEAR,,

GT,QB,NOR,1,16,3,1,2,FLIS,FLOS,,

**

**      VAI ARMAZENAR BLOCO

**

GB,S113,PRESE,CLEAR,CLOCK,J,K,Q,QB,,

NB,,
```

5.1.3 Circuito (fig. 5.4) e cartões de descrição do: 7474
"dual D type edge triggered flip-flop"

74S74 - DUAL D TYPE EDGE TRIGGERED FLIP - FLOP
PRESE
CLEAR
CLOCK
DADO
Q1
Q2
Q3
Q4
Q
QB

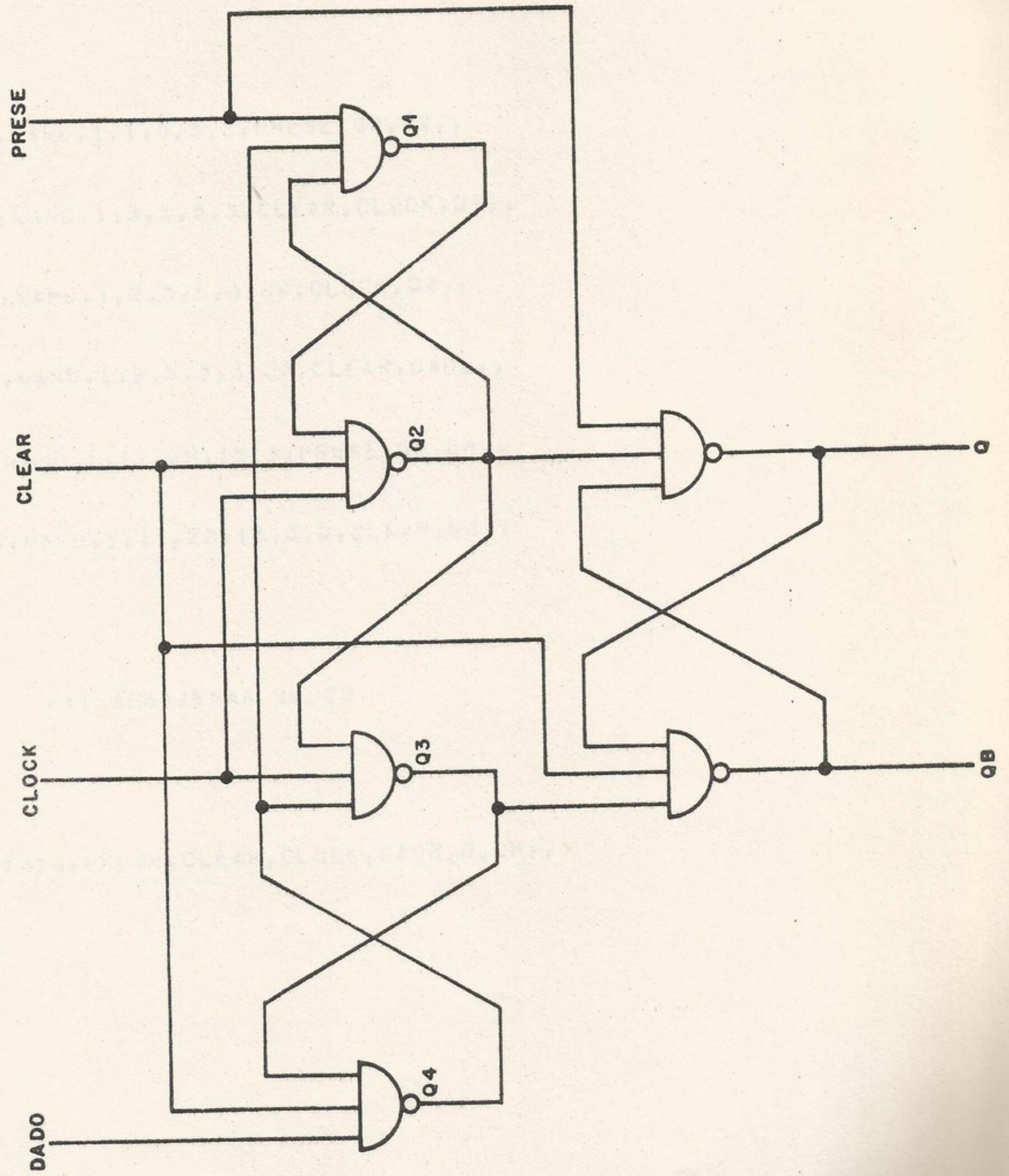



FIG. 5.4

```
**

**7474- DUAL D TYPE EDGE TRIGGERED FLIP-FLOP

**

GT,Q1,NAND,1,1,5,5,3,PRESE,Q4,Q2,,

GT,Q2,NAND,1,3,5,5,3,CLEAR,CLOCK,Q1,,

GT,Q3,NAND,1,2,5,5,3,Q2,CLOCK,Q4,,

GT,Q4,NAND,1,2,5,5,3,Q3,CLEAR,DADO,,

GT,Q,NAND,1,11,20,15,3,PRESE,Q2,QB,,

GT,QB,NAND,1,11,20,15,3,Q,CLEAR,Q3,,

**

**     VAI ARMAZENAR BLOCO

**

GB,B7474,PRESE,CLEAR,CLOCK,DADO,Q,QB,,

ND,,
```

5.1.4 Circuito (fig. 5.3) e cartões de descrição do: 74158 "quadruple two input selector/multiplexer"

74158 - QUADRUPLE 2- INPUT DATA SELECTOR / MULTIPLEXER
STR
SEL
B4
A4
B3
A3
B2
A2
B1
A1
SELA
SELB
BSTR
BB4
AA4
BB3
AA3
BB2
AA2
BB1
AA1
Y4
Y3
Y2
Y1

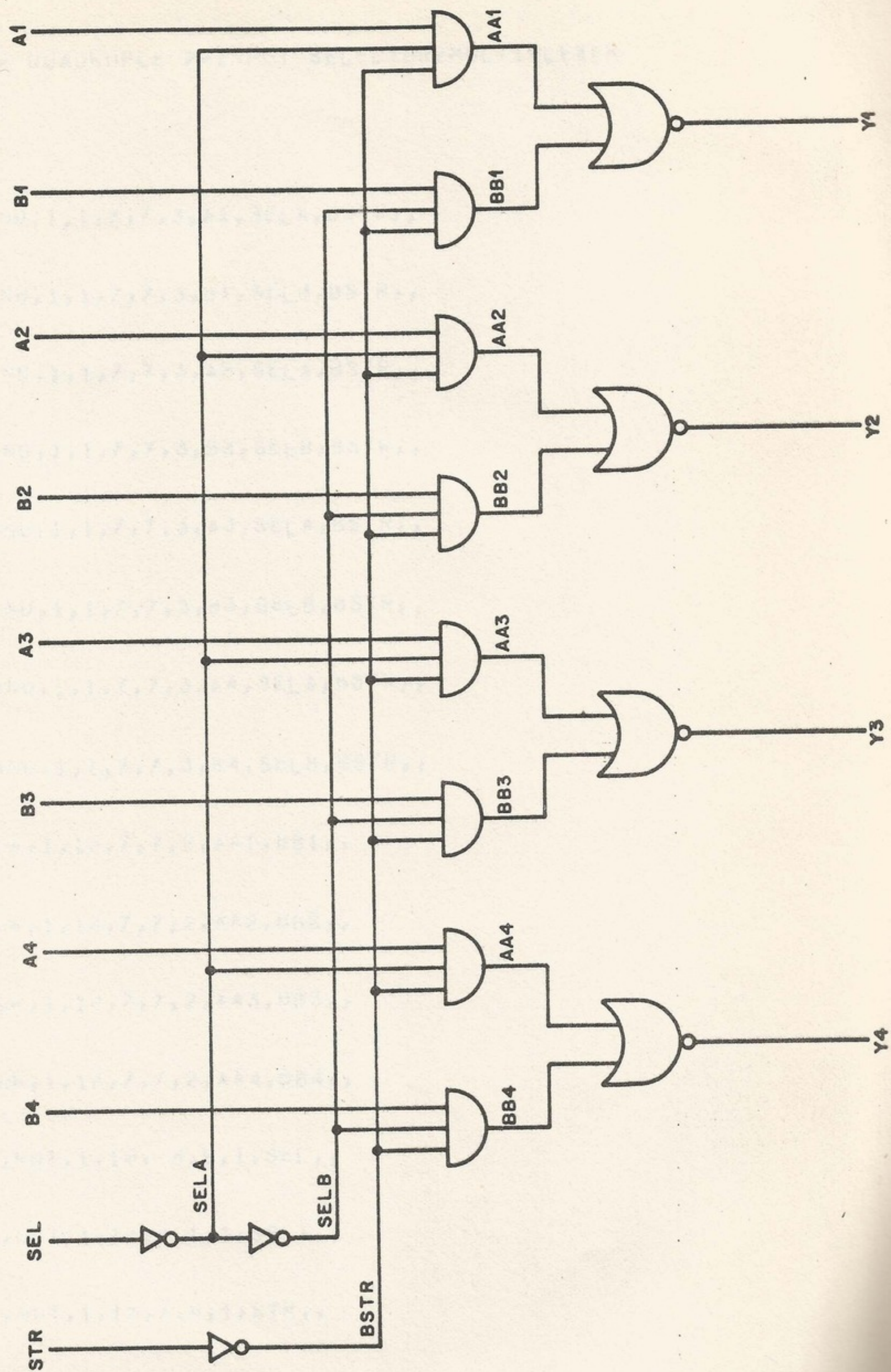



FIG. 5.3

```
**

**74158 - QUADRUPLE 2-INPUT SELECTOR/MULTIPLEXER

**

GT,AA1,AND,1,1,7,7,3,A1,SELA,BSTR,,

GT,BB1,AND,1,1,7,7,3,B1,SELB,BSTR,,

GT,AA2,AND,1,1,7,7,3,A2,SELA,BSTR,,

GT,BB2,AND,1,1,7,7,3,B2,SELB,BSTR,,

GT,AA3,AND,1,1,7,7,3,A3,SELA,BSTR,,

GT,BB3,AND,1,1,7,7,3,B3,SELB,BSTR,,

GT,AA4,AND,1,1,7,7,3,A4,SELA,BSTR,,

GT,BB4,AND.1,1,7,7,3,B4,SELB,BSTR,,

GT,Y1,NOR,1,10,7,7,2,AA1,BB1,,

GT,Y2,NOR,1,10,7,7,2,AA2,BB2,,

GT,Y3,NOR,1,10,7,7,2,AA3,BB3,,

GT,Y4,NOR,1,10,7,7,2,AA4,BB4,,

GT,SELA,NOT,1,10, 8,9,1,SEL,,

GT,SELB,NOT,1,10,4,1,1,SELA,,

GT,BSTR,NOT,1,10,7,6,1,STR,,

**

**      VAI ARMAZENAR BLOCO
```

```
**

GB,R158,STR,SEL,A1,A2,A3,A4,B1,B2,B3,B4,Y1,Y2,Y3,Y4,,

NB,,
```

5.1.5 Circuito (fig. 5.4) e cartões de descrição: 74S74
"dual D type edge triggered flip-flop"

```
**

**74S74- DUAL D- TYPE EDGE TRIGGERED FLIP- FLOP

**

GT,QS1,NAND,2,2,2,2,3,PRESE,QS4,QS2,,

GT,QS2,NAND,2,4,2,2,3,CLEAR,QS1,CLOCK,,

GT,QS3,NAND,1,2,2,2,3,CLOCK,QS2,QS4,,

GT,QS4,NAND,1,3,3,3,3,CLEAR,DADO,QS3,,

GT,Q,NAND,1,13,6,2,3,PRESE,QS2,QB,,

GT,QB,NAND,1,13,6,2,3,CLEAR,QS3,Q,,

**

**      VAI ARMAZENAR BLOCO

**

GB,S7474,PRESE,CLEAR,CLOCK,DADO,Q,QB,,

NB,,
```

5.1.6 Circuito (fig. 5.5) e cartões de descrição do:
9318
"eight input priority encoder"

9318 - EIGHT INPUT PRIORITY ENCODER
EBI
INB7
INB6
INB5
INB4
INB3
INB2
INB1
INBØ
IN7
IN6
IN5
IN4
IN3
IN2
IN1
INBB4
INBB6
INBB5
INBB2
EI
SA7
SA6
SA5
SA4
SA3
SA2
SA1
AB2
AB1
ABØ
GSB
EBØ

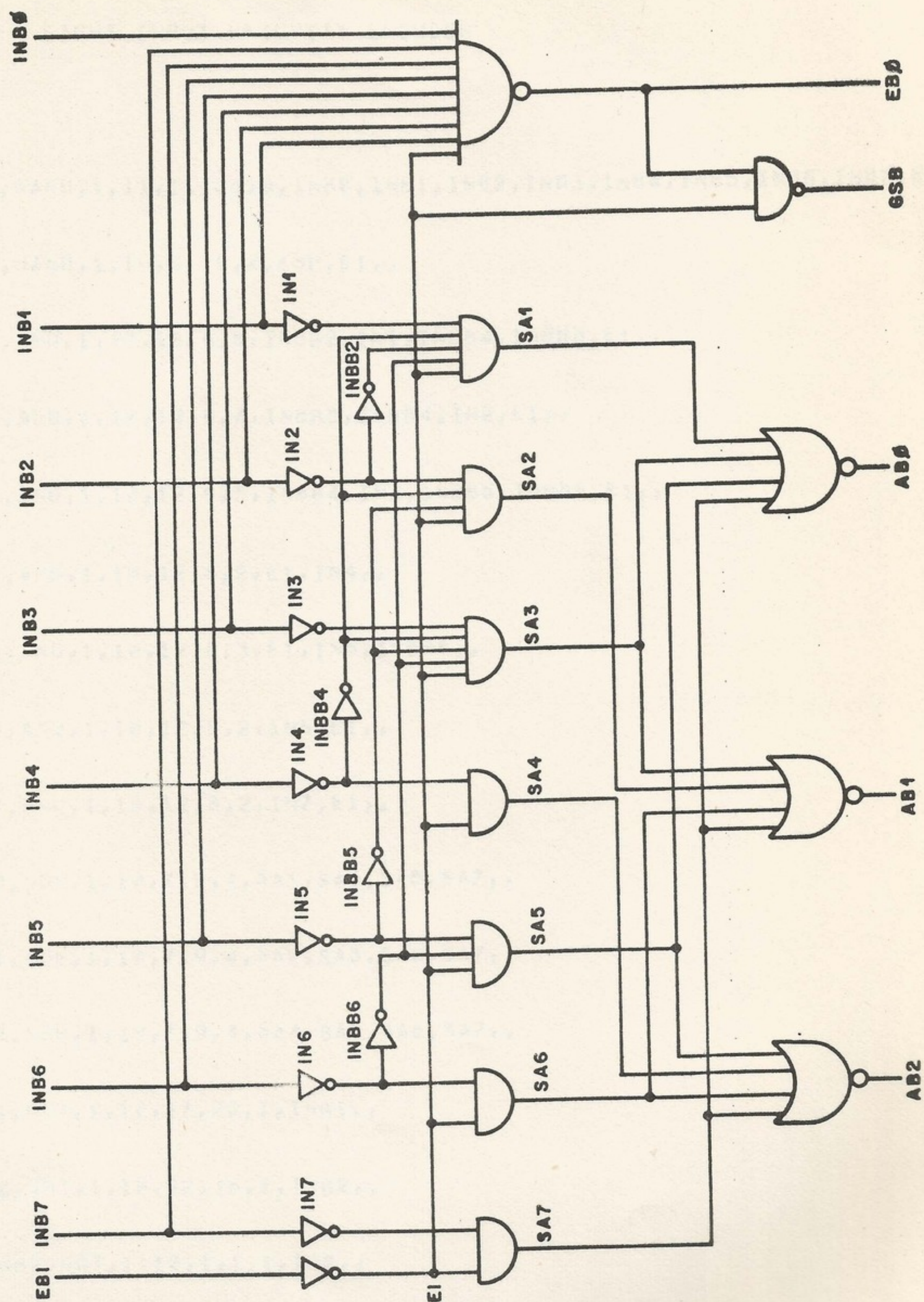



FIG. 5.5

```
**
**9318 - EIGHT INPUT PRIORITY ENCODER
**
GT,EB0,NAND,1,11,11,36,9,INB0,INB1,INB2,INB3,INB4,INB5,INB6,INB7,E1,,
GT,GSB,NAND,1,10,9,19,2,EB0,E1,,
GT,SA1,AND,1,10,12,8,5,INBB2,IN1,INBB4,INBB6,E1,,
GT,SA2,AND,1,10,12,8,4,INBB5,INBB4,IN2,E1,,
GT,SA3,AND,1,10,12,8,5,INBB4,IN3,INBB5,INBB6,E1,,
GT,SA4,AND,1,10,12,8,2,E1,IN4,,
GT,SA5,AND,1,10,12,8,3,E1,IN5,INBB6,,
GT,SA6,AND,1,10,12,8,2,IN6,E1,,
GT,SA7,AND,1,10,12,8,2,IN7,E1,,
GT,AB0,NOR,1,10,7,9,4,SA1,SA3,SA5,SA7,,
GT,AB1,NOR,1,10,7,9,4,SA2,SA3,SA6,SA7,,
GT,AB2,NOR,1,10,7,9,4,SA4,SA5,SA6,SA7,,
GT,IN1,NOT,1,10,17,23,1,INB1,,
GT,IN2,NOT,1,10,22,16,1,INB2,,
GT,INBB2,NOT,1,10,1,1,1,IN2,,
GT,IN3,NOT,1,10,17,23,1,INB3,,
GT,IN4,NOT,1,10,22,16,1,INB4,,
```

```
GT,INBB4,NOT,1,1Ø,1,1,1,IN4,,

GT,IN5,NOT,1,1Ø,22,16,1,INB5,,

GT,INBB5,NOT,1,1Ø,1,1,1,IN5,,

GT,IN6,NOT,1,1Ø,22,16,1,INB6,,

GT,INBB6,NOT,1,1Ø,1,1,1,IN6,,

GT,IN7,NOT,1,1Ø,17,23,1,INB7,,

GT,E1,NOT,2,1Ø,9,1Ø,1,EB1,,

**

**     VAI ARMAZENAR BLOCO

**

GB,B9318,EB1,INBØ,INB1,INB2,INB3,INB4,INB5,INB6,INB7,EBØ,GSB,ABØ,AB1,  *

  AB2,,

NB,,
```

5.1.7 Circuito (fig. 5.6) e cartões de descrição do:
74S175
"quadruple D type flip-flops with clear"

74S175 - QUADRUPLE D TYPE FLIP - FLOPS WITH CLEAR
Q11
Q12
Q13
Q14
Q1
QB1
D1
Q21
Q22
Q23
Q24
Q2
QB2
D2
CLEAR
CLEB
CLEBB
CLOCK
CLOB
CLOBB
Q31
Q32
Q33
Q34
Q3
QB3
D3
Q41
Q42
Q43
Q44
Q4
QB4
D4

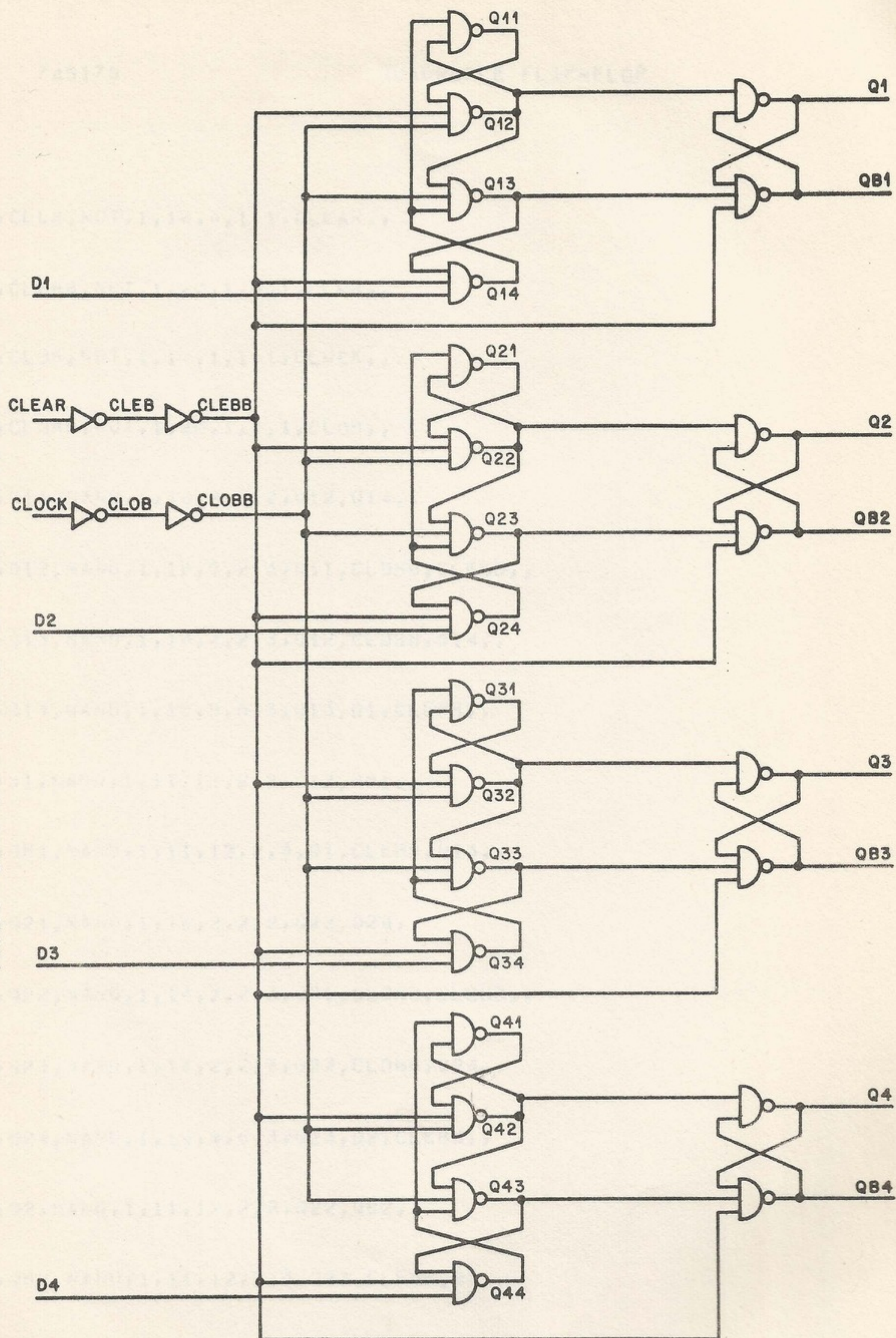



FIG. 5.6

```
**

**   74S175                    QUADRUPLE FLIP-FLOP

**

GT,CLEB,NOT,1,10,4,1,1,CLEAR,,

GT,CLEBB,NOT,1,20,1,4,1,CLEB,,

GT,CLOB,NOT,1,10,1,1,1,CLOCK,,

GT,CLOBB,NOT,1,20,1,1,1,CLOB,,

GT,Q11,NAND,1,10,2,2,2,Q12,Q14,,

GT,Q12,NAND,1,10,2,2,3,Q11,CLOBB,CLEBB,,

GT,Q13,NAND,1,10,2,2,3,Q12,CLOBB,Q14,,

GT,Q14,NAND,1,10,8,6,3,Q13,D1,CLEBB,,

GT,Q1,NAND,1,11,12,2,2,Q12,QB1,,

GT,QB1,NAND,1,11,12,2,3,Q1,CLEBB,Q13,,

GT,Q21,NAND,1,10,2,2,2,Q22,Q24,,

GT,Q22,NAND,1,10,2,2,3,Q21,CLOBB,CLEBB,,

GT,Q23,NAND,1,10,2,2,3,Q22,CLOBB,Q24,,

GT,Q24,NAND,1,10,8,6,3,Q23,D2,CLEBB,,

GT,Q2,NAND,1,11,12,2,2,Q22,QB2,,

GT,QB2,NAND,1,11,12,2,3,Q23,CLEBB,Q2,,

GT,Q31,NAND,1,10,2,2,2,Q32,Q34,,
```

```
GT,Q32,NAND,1,10,2,2,3,Q31,CLOBB,CLEBB,,

GT,Q33,NAND,1,10,2,2,3,Q32,CLOBB,Q34,,

GT,Q34,NAND,1,10,8,6,3,Q33,D3,CLEBB,,

GT,Q3,NAND,1,11,12,2,2,Q32,QB3,,

GT,QB3,NAND,1,11,12,2,3,Q3,CLEBB,Q33,,

GT,Q41,NAND,1,10,2,2,2,Q42,Q44,,

GT,Q42,NAND,1,10,2,2,3,Q41,CLOBB,CLEBB,,

GT,Q43,NAND,1,10,2,2,3,Q42,CLOBB,Q44,,

GT,Q44,NAND,1,10,8,6,3,Q43,D4,CLEBB,,

GT,Q4,NAND,1,11,12,2,2,Q42,QB4,,

GT,QB4,NAND,1,11,12,2,3,Q4,CLEBB,Q43,,

**

**      VAI ARMAZENAR BLOCO

**

GB,S175,CLEAR,CLOCK,D1,D2,D3,D4,Q1,QB1,Q2,QB2,Q3,QB3,Q4,QB4,,
```

5.1.8 Circuito (fig. 5.7) e cartões de descrição do: 3205 "high speed one out of eight binary decoder"

3205 - HIGH SPEED ONE OUT OF 8 BINARY DECODER
E3
EB2
EB1
E2
E1
A2
A1
AØ
ABØ
ABB2
ABB1
ABBØ
AB2
AB1
ABØ
S7
S6
S5
S4
S3
S2
S1
SØ

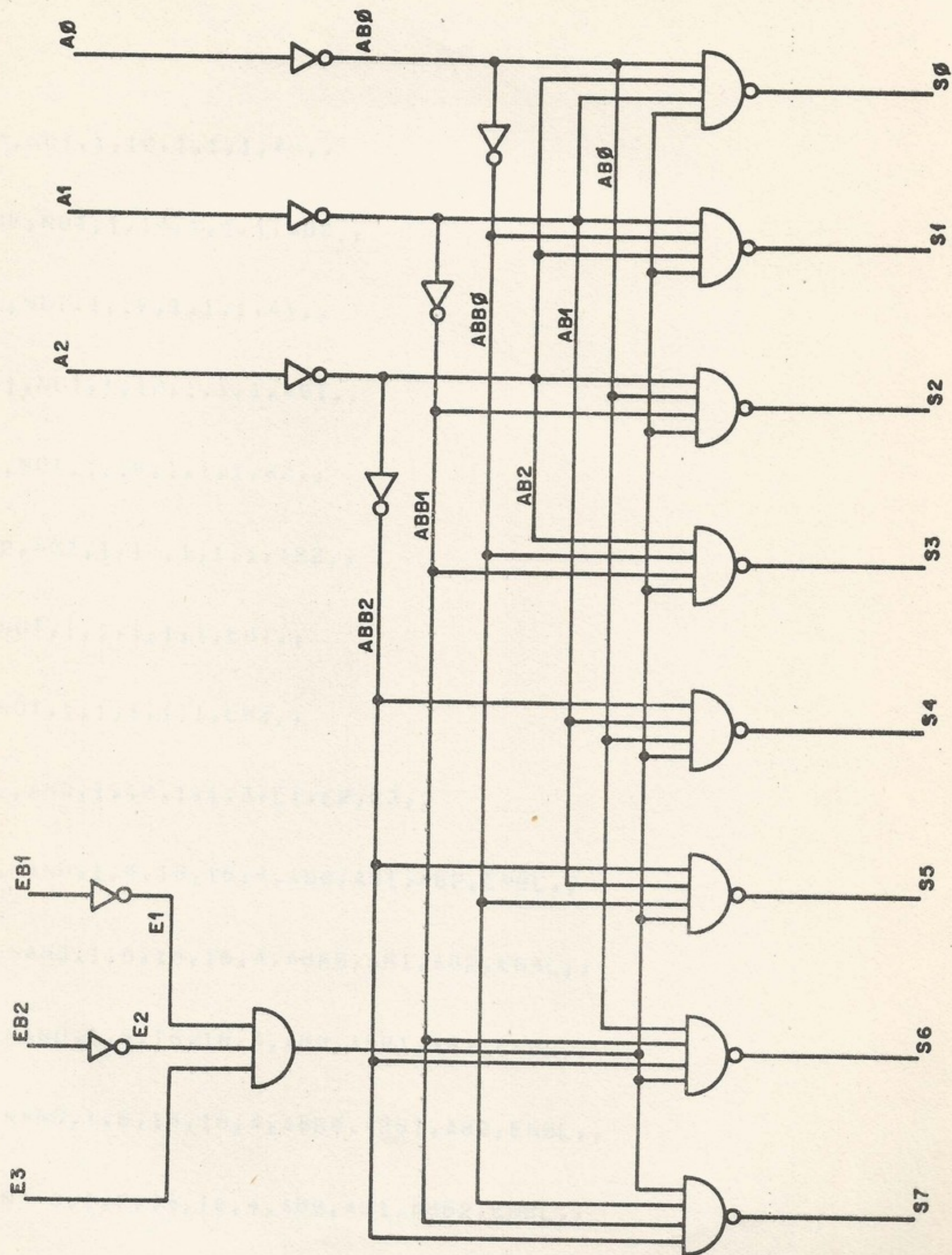
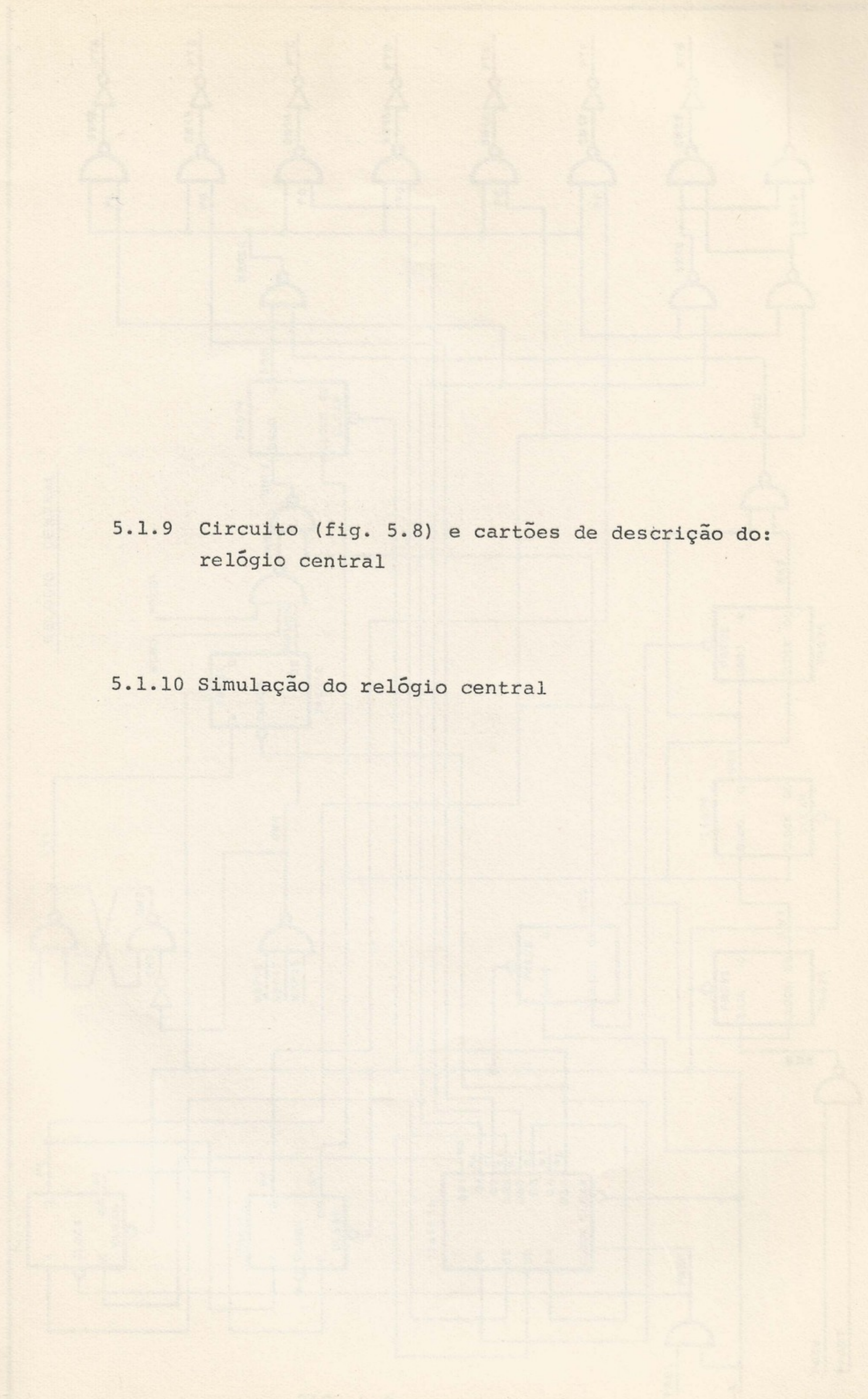



FIG. 5.7

```
**

**3205 - 1 OUT OF 8 BINARY DECODER

**

GT,AB0,NOT,1,10,1,1,1,A0,,

GT,ABB0,NOT,1,10,1,1,1,AB0,,

GT,AB1,NOT,1,10,1,1,1,A1,,

GT,ABB1,NOT,1,10,1,1,1,AB1,,

GT,AB2,NOT,1,10,1,1,1,A2,,

GT,ABB2,NOT,1,10,1,1,1,AB2,,

GT,E1,NOT,1,1,1,1,1,EB1,,

GT,E2,NOT,1,1,1,1,1,EB2,,

GT,ENBL,AND,1,10,1,1,3,E1,E2,E3,,

GT,SB0,NAND,1,6,16,16,4,AB0,AB1,AB2,ENBL,,

GT,SB1,NAND,1,6,16,16,4,ABB0,AB1,AB2,ENBL,,

GT,SB2,NAND,1,6,16,16,4,AB0,ABB1,AB2,ENBL,,

GT,SB3,NAND,1,6,16,16,4,ABB0,ABB1,AB2,ENBL,,

GT,SB4,NAND,1,6,16,16,4,AB0,AB1,ABB2,ENBL,,

GT,SB5,NAND,1,6,16,16,4,ABB0,AB1,ABB2,ENBL,,

GT,SB6,NAND,1,6,16,16,4,AB0,ABB1,ABB2,ENBL,,

GT,SB7,NAND,1,6,16,16,4,ABB0,ABB1,ABB2,ENBL,,
```

```
**

**      VAI ARMAZENAR BLOCO

**

GB,B3205,EB1,EB2,E3,A0,A1,A2,SB0,SB1,SB2,SB3,SB4,SB5,SB6,SB7,,

**

**

**  PEDIDO DE LISTAGEM DOS BLOCOS

**

**

LB,,
```

BIBLIOTECA DAS PASTILHAS EXISTENTES

NUM.BL.	BLOCO	ORIGEM	TAM.MFUN	TAM.MSIMB
1	B278	0	6	3
2	S113	9	2	2
3	B3205	13	4	3
4	B7474	20	2	2
5	B158	24	4	3
6	S7474	31	2	2
7	B9318	35	6	4
8	S175	45	7	4

EM CASO DE NECESSIDADE CONSULTE OS DADOS DE GERACAO DOS BLOCOS

5.1.9 Circuito (fig. 5.8) e cartões de descrição do: relógio central

5.1.10 Simulação do relógio central

RELÓGIO CENTRAL
74S112
J
Q
CLOCK
K
QB
CLEAR
PE
NE
PF
NF
NPRO
SN1
SN2
SN3
NCND
PDC31
NSP16
NSP47
NSP23
SN4
PRESE
NPRON
SN5
SN1Ø
74S74
DADO
SN11
NBREC
PA
PB
PC
PD
SN12
SN13
SN14
SN15
SN16
SN17
SN18
SN2Ø
SN19
PTA
PTB
PTC
PTD
PTE
PTF
NTØ
PTØ
74S175
D1
D2
D3
D4
QB1
Q1
Q2
QB2
QB3
Q3
Q4
QB4
NA
NB
NC
ND
NCU
PREL
PREF
NLAL
PBCU
PPART
SN6
SN7
SN8
SN9
PRCU

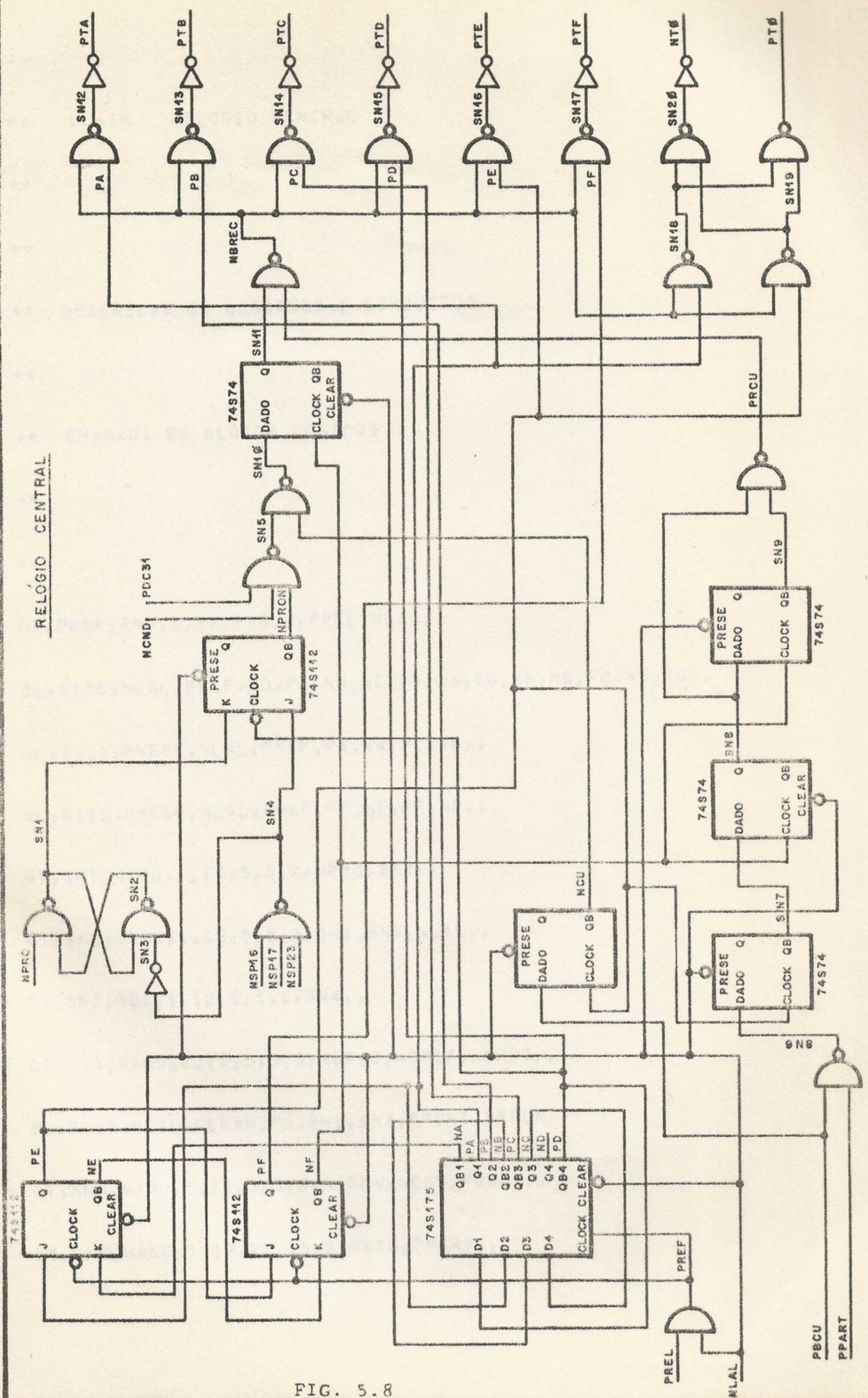



FIG. 5.8

```
**
** DL=18  RELOGIO CENTRAL
**
**
** DESCRICAO DE LIGACOES E CIRCUITOS
**
** CHAMADA DE BLOCOS LOGICOS
**
**
GT,PREF,AND,1,12,7,8,2,PREL,NLAL,,
BL,S175,NLAL,PREF,PD,PA,NB,NC,PA,NA,PB,NB,NC,PC,ND,PD,,
BL,S113,PRESE,NLAL,PREF,PA,NA,PE,NE,,
BL,S113,PRESE,NLAL,PREF,PE,NE,PF,NF,,
GT,SN1,NAND,1,12,5,5,2,NPRO,SN2,,
GT,SN2,NAND,1,12,5,5,3,SN1,SN3,NLAL,,
GT,SN3,NOT,1,12,5,5,1,SN4,,
GT,SN4,NAND,1,12,5,5,3,NSP16,NSP17,NSP23,,
BL,S113,NLAL,CLEAR,PD,SN1,SN4,PRONT,NPRON,,
GT,SN5,NAND,1,12,5,5,3,NPRON,NCND,PDC31,,.
GT,SN6,NAND,1,12,22,15,2,PBCU,PPART,,
```

```
BL,B7474,NLAL,CLEAR,PE,SN6,NUSA1,SN7,,

BL,B7474,NLAL,CLEAR,PE,PBCU,NUSA2,NCU,,

BL,S7474,PRESE,NLAL,NF,SN7,SN8,NUSA3,,

BL,S7474,NLAL,CLEAR,NF,SN8,NUSA4,SN9,,

GT,PRCU,NAND,1,12,5,5,2,SN8,SN9,,

GT,SN10,NAND,1,12,5,5,2,SN5,NCU,,

BL,S7474,PRESE,NLAL,NF,SN10,SN11,NUSA5,,

GT,NBREC,NAND,1,12,5,5,2,SN11,PRCU,,

GT,SN12,NAND,1,12,5,5,2,PA,NBREC,,

GT,PTA,NOT,1,12,8,7,1,SN12,,

GT,SN13,NAND,1,12,5,5,2,PB,NBREC,,

GT,PTB,NOT,1,12,8,7,1,SN13,,

GT,SN14,NAND,1,12,5,5,2,PC,NBREC,,

GT,PTC,NOT,1,12,8,7,1,SN14,,

GT,SN15,NAND,1,12,5,5,2,PD,NBREC,,

GT,PTD,NOT,1,12,8,7,1,SN15,,

GT,SN16,NAND,1,12,5,5,2,PE,NBREC,,

GT,PTE,NOT,1,12,8,7,1,SN16,,

GT,SN17,NAND,1,12,5,5,2,PF,NBREC,,

GT,PTF,NOT,1,12,8,7,1,SN17,,
```

```
GT,SN18,NAND,1,12,5,5,2,PA,NBREC,,

GT,SN19,NAND,1,12,5,5,2,PE,NBREC,,

GT,SN20,NAND,1,12,5,5,2,SN18,SN19,,

GT,NT0,NOT,1,12,8,7,1,SN20,,

GT,PT0,NAND,1,30,7,7,2,SN18,SN19,,

**

**DADOS DE TESTES

**

CR,NLAL,2000,1920,,

CL,PREL,50,25,,

GP,1,NPRO,2,300,200,400,500,,

GP,1,NSP16,2,120,380,700,,

CL,NSP17,1400,200,,

CL,NSP23,900,500,,

CR,PBCU,3000,2100,,

CR,PPART,3000,900,,

CL,PRESE,3000,2500,,

CL,CLEAR,3000,2500,,

GP,0,PDC31,2,120,380,100,300,300,,

GP,0,NCND,2,120,380,100,300,300,,

FI,,
```

SINAIS EXISTENTES

PREF	1	1	3
PREL	2	3	2000
NLAL	3	3	2000
PD	4	8	4
PA	5	8	6
NB	6	8	9
NC	7	8	9
NA	8	8	9
PB	9	8	9
PC	10	8	9
ND	11	8	10
@@@@@	12	9	9
@@@@@	13	9	8
@@@@@	14	9	9
@@@@@	15	9	12
@@@@@	16	9	9
@@@@@	17	9	7
@@@@@	18	9	8
@@@@@	19	9	8
@@@@@	20	9	9
@@@@@	21	9	7
@@@@@	22	9	8
@@@@@	23	9	8
@@@@@	24	9	9
@@@@@	25	9	7
@@@@@	26	9	8
@@@@@	27	9	8
@@@@@	28	9	9
@@@@@	29	9	7
@@@@@	30	9	8
@@@@@	31	9	8
PRESE	32	3	2000
PE	33	8	5
NE	34	8	11
@@@@@	35	9	8
@@@@@	36	9	8
@@@@@	37	9	9
@@@@@	38	9	9
@@@@@	39	9	9
@@@@@	40	9	9
PF	41	8	11
NF	42	8	3
@@@@@	43	9	8
@@@@@	44	9	8
@@@@@	45	9	9
@@@@@	46	9	9
@@@@@	47	9	9
@@@@@	48	9	9
SN1	49	1	10
NPRO	50	3	2000
SN2	51	1	11
SN3	52	1	11
SN4	53	1	10
NSP16	54	3	2000
NSP17	55	3	2000
NSP23	56	3	2000
CLEAR	57	3	2000
PRONT	58	8	12
NPRON	59	8	11

@@@@@	60	9	8
@@@@@	61	9	8
@@@@@	62	9	9
@@@@@	63	9	9
@@@@@	64	9	9
@@@@@	65	9	9
SN5	66	1	11
NCNU	67	3	2000
PDC31	68	3	2000
SN6	69	1	11
PBCU	70	3	2000
PPART	71	3	2000
NUSA1	72	8	10
SN7	73	8	9
@@@@@	74	9	0
@@@@@	75	9	0
@@@@@	76	9	0
@@@@@	77	9	0
NUSA2	78	8	10
NCU	79	8	9
@@@@@	80	9	0
@@@@@	81	9	0
@@@@@	82	9	0
@@@@@	83	9	0
SN8	84	8	10
NUSA3	85	8	12
@@@@@	86	9	0
@@@@@	87	9	0
@@@@@	88	9	0
@@@@@	89	9	0
NUSA4	90	8	12
SN9	91	8	11
@@@@@	92	9	0
@@@@@	93	9	0
@@@@@	94	9	0
@@@@@	95	9	0
PRCU	96	1	11
SN10	97	1	11
SN11	98	8	11
NUSA5	99	8	12
@@@@@	100	9	0
@@@@@	101	9	0
@@@@@	102	9	0
@@@@@	103	9	0
NBREC	104	1	4
SN12	105	1	11
PTA	106	1	12
SN13	107	1	11
PTB	108	1	12
SN14	109	1	11
PTC	110	1	12
SN15	111	1	11
PTD	112	1	12
SN16	113	1	11
PTE	114	1	12
SN17	115	1	11
PTF	116	1	12
SN18	117	1	10
SN19	118	1	10
SN20	119	1	11
NT0	120	1	12

```
PTØ          121          1          3Ø
SI,,
```

```
PV  PTF,PTE,PTD,PTC,PTØ,PTA,PTØ,NTØ,NBREC,PREL
   PV  PRONT
   TEMPO FINAL 3340
   FIM
```

	PTF	PTE	PTD	PTC	PTB	PTA	PTØ	NTØ	NBREC	PREL	PRONT
00000-	1	1	1	1	0	0	1	0	1	1	0
I	1	1	1	1	0	0	1	0	1	1	0
00001-	1	1	1	1	0	0	1	0	1	1	0
I	1	1	1	1	0	0	1	0	1	1	0
00002-	1	1	1	1	0	0	1	0	1	1	0
I	1	1	1	1	0	0	1	0	1	1	0
00003-	1	1	1	1	0	0	1	0	1	1	0
I	1	1	1	1	0	0	1	0	1	1	0
00004-	1	1	1	1	0	0	1	0	1	1	0
I	1	1	1	1	0	0	1	0	1	1	0
00005-	1	1	1	1	0	0	1	0	1	1	1
I	1	1	1	1	0	0	1	0	1	1	1
00006-	1	1	1	1	0	0	1	0	1	1	1
I	1	1	1	1	0	0	1	0	1	1	1
00007-	1	1	1	1	0	0	1	0	1	1	1
I	1	1	1	1	0	0	1	0	1	1	1
00008-	1	1	1	1	0	0	1	0	1	1	1
I	1	1	1	1	0	0	1	0	1	1	1
00009-	1	1	1	1	0	0	1	0	1	1	1
I	1	1	1	1	0	0	1	0	1	1	1
00010-	1	1	1	1	0	0	1	0	1	1	1
I	1	1	1	1	0	0	1	0	1	1	1
00011-	1	1	1	1	0	0	1	0	1	1	1
I	1	1	1	1	0	0	1	0	1	1	1
00012-	1	1	1	1	0	0	1	0	1	1	1
I	1	1	1	1	0	0	1	0	1	1	1
00013-	1	1	1	1	0	0	1	0	1	1	1
I	1	1	1	1	0	0	1	0	1	1	1
00014-	1	1	1	1	0	0	1	0	1	1	1
I	1	1	1	1	0	0	1	0	1	1	1
00015-	1	1	1	1	0	0	1	0	1	1	1
I	1	1	1	1	0	0	1	0	1	1	1
00016-	1	1	1	1	0	0	1	0	1	1	1
I	1	1	1	1	0	0	1	0	1	1	1
00017-	1	1	1	1	0	0	1	0	1	1	1
I	1	1	1	1	0	0	1	0	1	1	1
00018-	1	1	1	1	0	0	1	0	1	1	1
I	1	1	1	1	0	0	1	0	1	1	1
00019-	0	0	1	1	0	0	0	0	1	1	1
I	0	0	1	1	0	0	0	0	1	1	1
00020-	0	0	1	1	0	0	0	0	1	1	1
I	0	0	1	1	0	0	0	0	1	1	1
00021-	0	0	1	1	0	0	0	0	1	1	1
I	0	0	1	1	0	0	0	0	1	1	1
00022-	0	0	1	1	0	0	0	0	1	1	1
I	0	0	1	1	0	0	0	0	1	1	1
00023-	0	0	1	1	0	0	0	0	1	1	1
I	0	0	1	1	0	0	0	0	1	1	1
00024-	0	0	1	1	0	0	0	0	1	1	1
I	0	0	1	1	0	0	0	0	1	1	1
00025-	0	0	1	1	0	0	0	1	1	0	1
I	0	0	1	1	0	0	0	1	1	0	1
00026-	0	0	1	1	0	0	0	1	1	0	1
I	0	0	1	1	0	0	0	1	1	0	1
00027-	0	0	1	1	0	0	0	1	1	0	1
I	0	0	1	1	0	0	0	1	1	0	1
00028-	0	0	1	1	0	0	0	1	1	0	1
I	0	0	1	1	0	0	0	1	1	0	1
00029-	0	0	1	1	0	0	0	1	1	0	1
I	0	0	1	1	0	0	0	1	1	0	1
00030-	0	0	1	1	0	0	0	1	1	0	1
I	0	0	1	1	0	0	0	1	1	0	1
00031-	0	0	1	1	0	0	0	1	1	0	1
I	0	0	1	1	0	0	0	1	1	0	1

```
00032=  0  0  1  1  0  0  0  1  1  0  1
     I  0  0  1  1  0  0  0  1  1  0  1
00033=  0  0  1  1  0  0  0  1  1  0  1
     I  0  0  1  1  0  0  0  1  1  0  1
00034=  0  0  1  1  0  0  0  1  1  0  1
     I  0  0  1  1  0  0  0  1  1  0  1
00035=  0  0  1  1  0  0  0  1  1  0  1
     I  0  0  1  1  0  0  0  1  1  0  1
00036=  0  0  1  1  0  0  0  1  1  0  1
     I  0  0  1  1  0  0  0  1  1  0  1
00037=  0  0  1  1  0  0  0  1  1  0  1
     I  0  0  1  1  0  0  0  1  1  0  1
00038=  0  0  1  1  0  0  0  1  1  0  1
     I  0  0  1  1  0  0  0  1  1  0  '
00039=  0  0  1  1  0  0  0  1  1  0
     I  0  0  1  1  0  0  0  1  1  0
00040=  0  0  1  1  0  0  0  1  1  0
     I  0  0  1  1  0  0  0  1  1
00041=  0  0  1  1  0  0  0  1  '
     I  0  0  1  1  0  0  0  1
00042=  0  0  1  1  0  0  0  1
     I  0  0  1  1  0  0  0
00043=  0  0  1  1  0  0  0
     I  0  0  1  1  0  0  0
00044=  0  0  1  1  0  0  0
     I  0  0  1  1  0  0  0
00045=  0  0  1  1  0  0  0
     I  0  0  1  1  0  0  0
00046=  0  0  1  1  0  0  0
     I  0  0  1  1  0  0  '
00047=  0  0  1  1  0  0
     I  0  0  1  1  0  0
00048=  0  0  1  1  0  0
     I  0  0  1  1  0  0
00049=  0  0  1  1  0  0
     I  0  0  1  1  0  0
00050=  0  0  1  1  0  0
     I  0  0  1  1  0  0
00051=  0  0  1  1  0
     I  0  0  1  1  0
00052=  0  0  1  1  0
     I  0  0  '  '
00053=  0
     I  0
00054=  0
     I  0
00055=  0
     I  0
00056=
     '
```

5.1.11 Exemplo de problemas detetados com "fan-in" e "fan-out".

```
CL,R,7,3,,

CL,S,17,12,,

GT,JOAO,AND,1,1,1,1,1,S,,

GT,FLIP,NAND,2,1,1,1,2,JOAO,FLOP,,

GT,FLOP,NAND,1,1,1,1,2,R,FLIP,,

FI,,

PROBLEMAS DE FANIN-FANOUT COM BLOCO JOAO
PROBLEMAS DE FANIN-FANOUT COM BLOCO FLOP



  SINAIS EXISTENTES
R             1     3    1999
S             2     3    1999
JOAO          3     1      -1
FLIP          4     1       0
FLOP          5     1      -1
SI,,

ERROS SUPRIMEM EXECUCAO E GRAVACAO
```

```
CL,R,7,3,,

CL,S,17,12,,

GT,JOAO,AND,1,1,1,1,1,S,,

GT,FLIP,NAND,1,1,1,1,2,JOAO,FLOP,,

GT,FLOP,NAND,2,1,1,1,2,R,FLIP,,

FI,,

PROBLEMAS DE FANIN-FANOUT COM BLOCO FLIP




  SINAIS EXISTENTES
R             1        3     1998
S             2        3     1999
JOAO          3        1        0
FLIP          4        1       -1
FLOP          5        1        0
SI,,

ERROS SUPRIMEM EXECUCAO E GRAVACAO
```

5.1..12 Exemplos de outros tipos de erros detetados.

```
CL,S,10,5,,

CR,R,10,5,,

**

**NAND FLIP FLOP

**

GT,FLIP,NAND,1,1,1,1,2,S,FLOP,,

GT,FLOPFLOP,NAND,1,1,1,1,2,R,FLIP,,

FASE DE COMPILACAO--ERRO NUM.    31    COLUNA=  9
FI,,
```

```
CL,S,10,5,,

CR,R,10,5,,

**

**NAND FLIP FLOP

**

GT,FLIP,NAND,1,1,1,1,2,S,FLOP,,

GT,FLOP,NAND,1,1,1,1,3,R,FLIP,,


FASE DE COMPILACAO--ERRO NUM.   10    COLUNA= 31
FI,,
```

6. Observações Finais

6. Observações Finais

O programa para simulação em nível de portas lógicas mostrou-se, após os testes realizados e vários casos práticos processados, bastante eficiente.

Inúmeros problemas que escapam ao projetista puderam ser detetados pelo simulador e levaram a um reestudo do projeto lógico e sua correção.

O algorítmo utilizado mostrou-se bastante flexível pois foi aplicado indistintamente tanto na simulação em nível de portas lógicas como também no programa simulador a nível de registros, tendo-se obtido os resultados esperados nas duas aplicações. Com essa observação, pode-se antecipar a utilização da mesma filosofia do algorítmo em outras aplicações.

Os atrasos envolvidos na simulação foram limitados apenas aos valores fornecidos pelo usuário mas são em geral os próprios valores nominais fornecidos pelo fabricante. O programa simulador permite a análise de condições de funcionamento de um circuito atráves da variação dos valores de atraso, de seus blocos dentro dos limites de máximo e mínimo especificados pelo fabricante.

Essa análise poderia ser automatizada atráves de algumas modificações na estrutura de dados do simulador bem como no programa que trabalha com a simulação propriamente dita.

Essa alternativa levaria a um estudo exaustivo do comportamento do sistema sendo simulado.

Entretanto, os valores de atrasos utilizados dizendo respeito aos tempos "TD01" e "TD10", associados com sua escolha atráves da rotina de simulação do bloco e do comportamento do programa principal, permitem detetar durante o processo de simulação a maior parte dos eventuais problemas que costumam escapar a atenção do mais experimentado projetista.

O algorítmo para verificação dos problemas de "fan-in" e "fan-out" apesar de simples ajudou na solução de problema de carga de blocos lógicos.

Uma melhoria poderia ser acrescentada nos blocos de controle, dotando-os de características de maior poder e eficiência. Assim poderia ser acrescentado alguns blocos que exercem funções mais globais, como por exemplo: substituir a descrição de um determinado bloco pela descrição que se segue ao controle, inserir um bloco cuja descrição aparece em seguida ao bloco simulado, retirar um determinado bloco da estrutura, substituir sinal de estímulo por outro cuja descrição se segue, etc.

Esses recursos adicionais serão incorporados ao sistema, que deverá então basear-se em um computador de maior porte.

B I B L I O G R A F I A

1. BREUER, M.A. - Design automation of digital systems. Englewood Cliffs, Prentice-Hall,1972. v.1 -Theory and techniques.

2. STEPHENSON, R.E. - Computers simulation for engineers. Utah, University of Utah, 1971.

3. SZYGENDA, S.A; ROUSE, D.M & THOMPSON, E.D. - "A model and implementation of a universal time delay simulator for large digital nets." In: SPRIN JOINT COMPUTER CONFERENCE, University of Missouri, 1970. - -Proceedings. Roela, Missouri, 1970, p. 207 - 216

4. BREUER, M.A. -"Recent developments in the automated design and analysis of digital systems".Proceedings of the IEEE, 60(1): 12-27, jan.1972

5. LAKE, D.W. -"Logic simulation in digital systems".Computer Design, 9(5): 77-83, May 1970.

6. CHU, Y. - Introduction to computer organization.Englewood Cliffs, Prentice-Hall, 1970.

7. TUNG, L.H. - A unit delay logic timing simulator. IBM Systems Development Division, Endicott Laboratory, TD 01.509. Endicott, IBM, 1969.

8. SCHEFF, B. H. -"A machine aids system for digital designers. "Computer Design, 8(10): 76-81, oct. 1969.

9. HOWIE, H.R., TAVAN, R.M. - The on-line logical simulation system. Cambridge, M.I.T., Charles Stark Draper Laboratory.

10. BERETEVAS, T., LIU, C.H. & TAYLOR, R.L. - "A general purpose design automation file systems". In: DESIGN AUTOMATION WORKSHOP. 4th, 1966. - Proceedings.

11. McCLUSKEY, E.J. - Introduction to the theory of switching circuits. New York, McGraw-Hill, 1965.

12. TEXAS INSTRUMENTS INC. - The TTL data Book for design engineers. 1971.

13. FAIRCHILD SEMICONDUCTOR - TTL data book. California, Fairchild, 1972.

14. HEWLETT PACKARD - Disc operating system HP 02116-91748. New Jersey, Hewlett Packard, 1969.